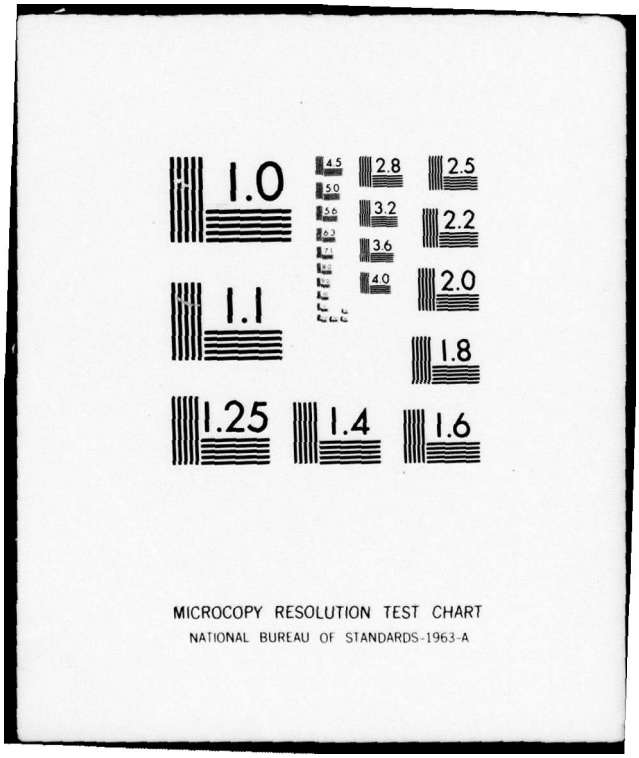
AD-A039 328 FRANKFORD ARSENAL PHILADELPHIA PA F/G 9/2
DIGITAL LANGUAGE SYNTAX.(U)
FEB 77 R L GAUTHIER
UNCLASSIFIED FA-FCF-1-77 NL

| OF |
AD
A039328

END
DATE
FILMED
5-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

REPORT FCF-1-77

FINAL REPORT

DIGITAL LANGUAGE SYNTAX

February 1977

U.S. ARMY ARMAMENT COMMAND
FRANKFORD ARSENAL
PHILADELPHIA, PENNSYLVANIA 19137

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: FCF-1-77	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Digital Language Syntax	5. TYPE OF REPORT & PERIOD COVERED: Final Report
	6. PERFORMING ORG. REPORT NUMBER: 738-2
7. AUTHOR(s): R. L. Gauthier	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: RLG Associates, Inc. 11250 Roger Bacon Drive Reston, VA 22090	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Frankford Arsenal Phila., PA 19137	12. REPORT DATE: Feb 77
	13. NUMBER OF PAGES: 77
14. MONITORING AGENCY NAME & ADDRESS *(If different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Operational Performance Analysis Language (OPAL).
Digital Testing
Dynamic Testing
Virtual Resource

142 720

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report represents a study of digital testing and the integration of digital constructs into OPAL. Identified are the language elements that must be modified or added to OPAL to incorporate digital testing. Examples using these constructs are discussed and change sheets for the OPAL specification are included.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

INSTRUCTIONS FOR PREPARATION OF REPORT DOCUMENTATION PAGE

RESPONSIBILITY. The controlling DoD office will be responsible for completion of the Report Documentation Page, DD Form 1473, in all technical reports prepared by or for DoD organizations.

CLASSIFICATION. Since this Report Documentation Page, DD Form 1473, is used in preparing announcements, bibliographies, and data banks, it should be unclassified if possible. If a classification is required, identify the classified items on the page by the appropriate symbol.

COMPLETION GUIDE

General. Make Blocks 1, 4, 5, 6, 7, 11, 13, 15, and 16 agree with the corresponding information on the report cover. Leave Blocks 2 and 3 blank.

Block 1. Report Number. Enter the unique alphanumeric report number shown on the cover.

Block 2. Government Accession No. Leave Blank. This space is for use by the Defense Documentation Center.

Block 3. Recipient's Catalog Number. Leave blank. This space is for the use of the report recipient to assist in future retrieval of the document.

Block 4. Title and Subtitle. Enter the title in all capital letters exactly as it appears on the publication. Titles should be unclassified whenever possible. Write out the English equivalent for Greek letters and mathematical symbols in the title *(see "Abstracting Scientific and Technical Reports of Defense-sponsored RDT/E," AD-667 000)*. If the report has a subtitle, this subtitle should follow the main title, be separated by a comma or semicolon if appropriate, and be initially capitalized. If a publication has a title in a foreign language, translate the title into English and follow the English translation with the title in the original language. Make every effort to simplify the title before publication.

Block 5. Type of Report and Period Covered. Indicate here whether report is interim, final, etc., and, if applicable, inclusive dates of period covered, such as the life of a contract covered in a final contractor report.

Block 6. Performing Organization Report Number. Only numbers other than the official report number shown in Block 1, such as series numbers for in-house reports or a contractor/grantee number assigned by him, will be placed in this space. If no such numbers are used, leave this space blank.

Block 7. Author*(s)*. Include corresponding information from the report cover. Give the name*(s)* of the author*(s)* in conventional order *(for example, John R. Doe or, if author prefers, J. Robert Doe)*. In addition, list the affiliation of an author if it differs from that of the performing organization.

Block 8. Contract or Grant Number*(s)*. For a contractor or grantee report, enter the complete contract or grant number*(s)* under which the work reported was accomplished. Leave blank in in-house reports.

Block 9. Performing Organization Name and Address. For in-house reports enter the name and address, including office symbol, of the performing activity. For contractor or grantee reports enter the name and address of the contractor or grantee who prepared the report and identify the appropriate corporate division, school, laboratory, etc., of the author. List city, state, and ZIP Code.

Block 10. Program Element, Project, Task Area, and Work Unit Numbers. Enter here the number code from the applicable Department of Defense form, such as the DD Form 1498, "Research and Technology Work Unit Summary" or the DD Form 1634, "Research and Development Planning Summary," which identifies the program element, project, task area, and work unit or equivalent under which the work was authorized.

Block 11. Controlling Office Name and Address. Enter the full, official name and address, including office symbol, of the controlling office. *(Equates to funding/sponsoring agency. For definition see DoD Directive 5200.20, "Distribution Statements on Technical Documents.")*

Block 12. Report Date. Enter here the day, month, and year or month and year as shown on the cover.

Block 13. Number of Pages. Enter the total number of pages.

Block 14. Monitoring Agency Name and Address *(if different from Controlling Office)*. For use when the controlling or funding office does not directly administer a project, contract, or grant, but delegates the administrative responsibility to another organization.

Blocks 15 & 15a. Security Classification of the Report: Declassification/Downgrading Schedule of the Report. Enter in 15 the highest classification of the report. If appropriate, enter in 15a the declassification/downgrading schedule of the report, using the abbreviations for declassification/downgrading schedules listed in paragraph 4-207 of DoD 5200.1-R.

Block 16. Distribution Statement of the Report. Insert here the applicable distribution statement of the report from DoD Directive 5200.20, "Distribution Statements on Technical Documents."

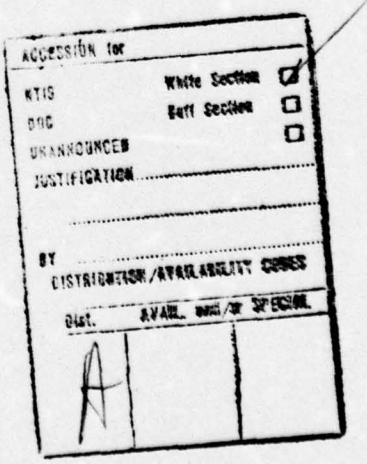
Block 17. Distribution Statement *(of the abstract entered in Block 20, if different from the distribution statement of the report)*. Insert here the applicable distribution statement of the abstract from DoD Directive 5200.20, "Distribution Statements on Technical Documents."

Block 18. Supplementary Notes. Enter information not included elsewhere but useful, such as: Prepared in cooperation with . . . Translation of *(or by)* . . . Presented at conference of . . . To be published in . . .

Block 19. Key Words. Select terms or short phrases that identify the principal subjects covered in the report, and are sufficiently specific and precise to be used as index entries for cataloging, conforming to standard terminology. The DoD "Thesaurus of Engineering and Scientific Terms" *(TEST)*, AD-672 000, can be helpful.

Block 20. Abstract. The abstract should be a brief *(not to exceed 200 words)* factual summary of the most significant information contained in the report. If possible, the abstract of a classified report should be unclassified and the abstract to an unclassified report should consist of publicly- releasable information. If the report contains a significant bibliography or literature survey, mention it here. For information on preparing abstracts see "Abstracting Scientific and Technical Reports of Defense-Sponsored RDT&E,"

TABLE OF CONTENTS



I. INTRODUCTION

1. Background

Digital testing differs from analogue testing primarily in that a multitude of resources need to be activated simultaneously. In addition, for high-speed (dynamic) testing, precise time control is required between the activation of events from different resources. Neither of these characteristics is available in the current version of OPAL.

OPAL is a highly structured language, and by virtue of its modular syntax, imposes few restrictions on syntactical constructs. It was, therefore, decided that the recommended "digital" constructs would, in general, not be tailored only to "digital" testing in the form of a "template language" but would provide the capability to describe the requirements in general terms. Therefore, any appropriate signal oriented statements could be utilized within the new structures.

2. Concepts and Reasoning

a) The introduction of sets (single dimensional arrays) of resources with similar or identical characteristics provides the mechanism by which multiple resources can be activated simultaneously or within a controlled time environment.

The resource sets may be fully partitioned in a structured fashion. This will enable sub-sets of any desired level to provide the vehicle to specify special limitations and connections to those sub-sets.

b) Resource sets may be referenced as an entire set, or as individual components referenced by index selection, or as sub-sets referenced by a structured notational form which relates each sub-set to its base set.

c) Functional testing where interstatement timing is not critical will be performed utilizing the already existing itera-

tive structures and modified CHANGE and READ statements (LOOP, FOR, etc.).

d) Dynamic testing will be performed by the introduction of a new time-control statement (RUN) which contains stimulus and response iterative clauses. In the interest of ensuring that high speed and precise timing can be achieved, these structures are restricted. They require that data analysis be deferred until after invocation of the RUN statement.

e) Although the current BITSTRING data mode and associated operators provide all the necessary basic tools to analyze response data, it was felt that the process would be both error prone and unnecessarily copious. As a result, the DECLARE statement for strings was extended to enable fully partitioned structured substrings to be specified. Reference of those substrings utilizes a notational form consistent with that utilized to reference resource sub-sets. This approach was selected partially to maintain language consistency but also to emphasize the relationship between bits within strings and components within resource sets.

f) The mapping of bits within strings to resource sets is implied by the ordinal positions of bits and resource components within the DECLARE and REQUIRE statements, respectively. For example, if the bitstring S is declared as having 10 bits and the digital resource set, D, of 10 components, the most significant bit (bit-1) of S is associated with the resource component D(1), the next most significant bit of S (bit-2) is associated with the resource component D(2), etc., if the data from S is used as a STATE modifier for the resource set D.

g) To enable any characteristic to represent logical data the suffix LOGIC will be used with any applicable noun. The use of the suffix implicitly defines that there is a logical state bit-string modifier associated with the resource. To define the logical states requires that the modifier appear twice, with 0 and 1 suffixes, in both the resource definition and the appropriate executable statements.

h) To completely describe the limitations of a digital logic resource, both changes and additions were made to the modifier set. These are summarized below.

1) STATE — modifier to describe the logical 'state' of a digital logic resource set.

2) 0,1 — modifier suffixes to be used with the modifiers defining the logical state to describe the analogue characteristics which represent a 0 and 1 respectively in the STATE bitstring.

i) The use of a digital logic resource may require the use of additional modifiers to the set associated with the noun, these are summarized below:

1) DATA_RATE — Describes, for sources, the rate at which CHANGEs can occur, and for sensors the rate at which READings can be made, normally has an HZ dimension. In general, this modifier must be CONTROLable, and is programmed by the at_rate clause in the CHANGE and READ statements, for source and sensor resources respectively. However, if the usage within an entire test program requires that the value is constant, it may appear as a CAPABILITY modifier.

2) DATA_WIDTH — Used only with source resources and describes the pulse width, in units of time for a single 'bit' of data. In general, this modifier must be CONTROLable and is programmed by the for-time clause appearing in the change statement. However, if the usage within an entire test program requires that the value is constant, it may appear as a CAPABILITY modifier.

3) BUFFER_SIZE — As later described in this report, the model of a digital resource for dynamic testing includes a memory buffer. The dynamic testing construct proposed describes the number of 'bits' transmitted or received, at the specified rate, in a contiguous stream. In order to

determine if a specific actual resource can satisfy the virtual resource, it is necessary for the maximum number of bits issued in a continuous stream, at any point within the entire test program, be specified as a CAPABILITY BUFFER_SIZE modifier in the virtual resource REQUIRE statement. It should be understood that if the actual resource BUFFER_SIZE is less than the virtual, it does not necessarily preclude the possibility of the REQUIREMENT being satisfied. This is so since the maximum REQUIRE DATA_RATE may be such that buffering is not required.

j) Resources which have a single input or output port need not have a connection point identified since it is obviously redundant. In addition, it simplifies the writing of digital test statements since such statements frequently satisfy the criteria of a single port.

k) The modifications to the SPECIFY statement introduce test point arrays, henceforth referred to as sets. These arrays are limited to a single dimension but unlimited structured subsets are permitted. These changes were effected to provide a shorthand notation for providing the connection between a set or sub-set of resources and a set or sub-set of test points.

3. Dynamic Testing

Dynamic testing implies that stimulus and response data rates will be at the normal or maximum operating speed of the unit under test. In general, with the current generation of computer controlled test devices, such test procedures cannot be achieved by direct control of the test interface from the CPU. I/O bus data rates cannot function at sufficiently high speeds. The technique most commonly employed to run dynamic tests is to provide the test interface with a high speed buffer which can, for stimuli, be loaded with data from the CPU memory at relatively low speeds. The data from the high speed buffer is subsequently clocked to the test interface. For response data, the test interface is strobed at high speed and the data is temporarily stored in the high speed buffer. When sufficient data has been gathered, or when the buffer memory is full, the data is transferred from the high speed buffer to the CPU memory at relatively low speeds. A block diagram of a typical system follows.

There are a number of very significant constraints which result from a system of this nature and it is our opinion that they should be considered in the design of the dynamic testing language constructs. Those constraints are enumerated below:

a) Analysis of response data cannot be invoked during execution of the test. It is realized that there are, at this time, a small percentage of testers which can perform limited analysis "on the fly". This limited analysis usually includes the ability to mask the results and perhaps compare them with an expected response and to generate an interrupt when a mismatch occurs. It was described in our interim report that such esoteric features would not be included in this initial revision to the language.

b) However large the high-speed buffer memory(s) is, it is finite, and will require filling or flushing from or to the CPU memory at various intervals during lengthy test sequences. Thus, the specified clock frequency can only be achieved for a limited number of patterns. In general, discontinuities of this nature are of little importance. However, when testing charge storage devices, minimum frequency requirements may be imposed by the very nature of those devices. It is feasible that this problem could be resolved by providing a high frequency 'refresh' of the last pattern issued during that time that the stimulus buffer was being filled from the CPU memory. However, such a technique does not appear to be widely employed and the alternative of allowing the test programmer to control discontinuity points was considered a viable solution.

c) The bounds of the data vectors must be defined prior to execution of the specific test and the iterator control variable must not change within a single loop of the stimulus-delay-response sequence.

II. REQUIRE Statement

A REQUIRE statement describes a virtual resource needed to conduct a test of a UUT and assumed to be available.

1. Syntax

```
require-stmt    ::=   REQUIRE resource-name,... { test-resource  } ;
                                                { clock-resource }
                                                { i/o-resource   }

resource-name   :=    name

test-resource   ::=   { noun sig-modifier  { SOURCE } }
                      {                    { SENSOR } } test-resource-desc
                      { IMPEDANCE sig-modifier LOAD  }

test-resource-desc  ::= [test-resource-structure] test-resource-spec

                        [CNX req-conn]

test-resource-spec  ::=  WITH [control-limitation][capability-limitation]

                         [limit-limitation]

control-limitation  ::= CONTROL req-limitation,...

capability-limitation  ::=  CAPABILITY req-limitation,...

limit-limitation    ::=  LIMIT req-limitation,...

req-limitation  ::=   { sig-limitation  }
                      { test-limitation }
                      { thru-limitation }

sig-limitation  ::=    sig-modifier req-mod-desc
```

```
test-limitation ::=     test-modifier req-mod-desc

thru-limitation ::=     thru-modifier req-mod-desc

req-mod-desc    ::=   { req-mod-value }
                      { req-mod-range }

req-mod-value   ::=     eq   { numerical-constant,...[accuracy]          }
                             { (numerical-constant accuracy),...         }

req-mod-range   ::=     ({gt-lt numeric-constant} |

                          {eq numeric-constant TO numeric-constant

                            [BY numeric-constant])}[accuracy]

gt-lt           ::=   { > }
                      { >= }
                      { < }
                      { <= }

eq              ::=     =

req-conn        ::=     (pin-descriptor = resource-pin),...

test-resource-structure  ::=  SET (set-clause,...)

set-clause      ::=   { integer-constant                          }  [after-clause]
                      { resource-name [(set-clause,...)]          }

                      [test-resource-spec][CNX req-conn]

after-clause    ::=     AFTER_DELAY time-clause
```

```
time-clause     ::=  { { gt-lt } time-interval                    }
                     { { eq    }                                  }
                     {                                            }
                     { time-interval TO time-interval             }

sig-modifier    ::=  name

noun            ::=  name[noun-suffix]

modifier        ::=  name[modifier-suffix]

noun-suffix     ::=  _name

modifier-suffix ::=  _name
```

2. Constraints

Pin descriptors and modifiers must be appropriate to the indicated noun. Resource_pins and resource_names witnin a single set_clause and test_resource must be unique.

3. Examples

a) REQUIRE T2 TIME INTERVAL CLOCK;

b)
```
REQUIRE ACS AC VOLTAGE SOURCE
        WITH CONTROL VOLTAGE=1 VOLT TO 10 VOLT BY 100 M*VOLT,
                     FREQ=50 HZ TO 400 HZ
             CAPABILITY CURRENT=1 AMP
             LIMIT CURRENT=2 AMP
        CNX HI=A, LO=B;
```

c)
```
REQUIRE ACVM AC VOLTAGE SENSOR
        WITH CAPABILITY VOLTAGE<=450 VOLT,
                        FREQ=50 HZ TO 440 HZ
        CNX HI=A, LO=B, LO=G;
```

d)
```
REQUIRE LD LOAD IMPEDANCE
        WITH CAPABILITY IMPEDANCE=1000 OHM
             LIMIT      CURRENT_TRMS=1 AMP
```

```
        CNX P1=HI, P2=LO;
```

e)
```
REQUIRE DSSD DC_LOGIC VOLTAGE SOURCE SET
        (TTL(8) WITH CONTROL VOLTAGE_1=4.7 VOLT,
                             VOLTAGE_0=0 VOLT
                  CAPABILITY CURRENT=100 M*AMP,
         SLL(4) WITH CONTROL VOLTAGE_1=-1.2 VOLT ,
                             VOLTAGE_0=5.8 VOLT
        )  WITH LIMIT RISE_TIME < 100 N*SEC
           CNX TRU=D, COMP=G;
```

4. Semantics

These semantics define only the structural concepts (i.e., Sets) for which this organization is responsible. The remainder of the statement semantics must be appended.

A single identifier (resource_name) may refer to a set of identical or similar resources. Any action referencing such an identifier implies that the action to each component of the set is to be performed concurrently. The number of components within the set is defined by the sum of the integer-constants which appear within the construct. Fully partitioned subsetting of contiguous groups of components, with or without names, may be specified by the recursive nature of the set-clause construct.

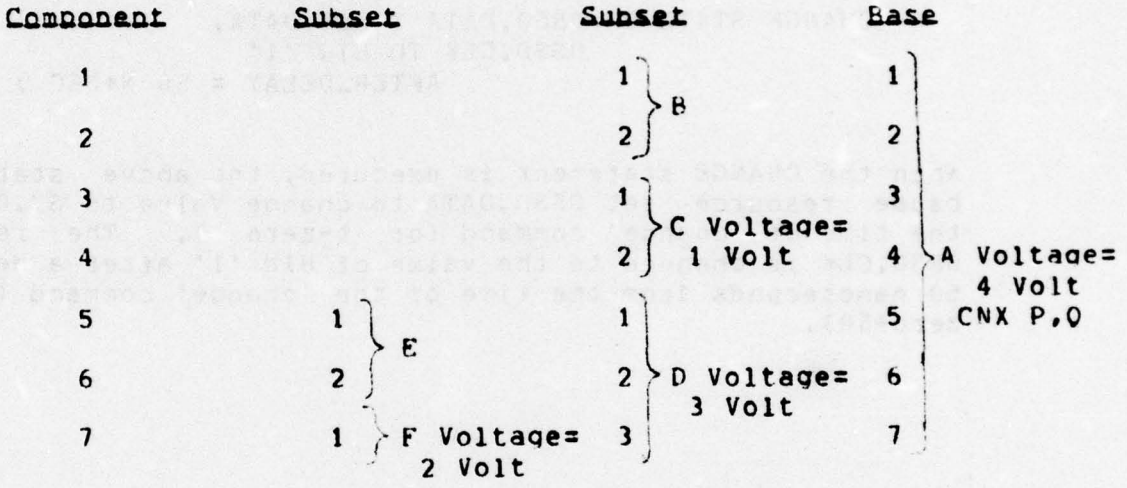
Test resource specifications (test_resource_spec) and connections (CNX req_conn) are local to the scope in which they are defined and global to any subsets within that structure with the following exception:

> If they are respecified within a lower level structure, that respecification becomes valid for the scope in which it is defined. Of course this rule applies recursively.

The following example shows the scoping:

```
REQUIRE  A DC VOLTAGE SOURCE SET
          (B(2),C(2) WITH CONTROL VOLTAGE=1 VOLT,
           D(E(2),F WITH CONTROL VOLTAGE=2 VOLT)
             WITH CONTROL VOLTAGE=3 VOLT)
```

```
) WITH CONTROL VOLTAGE=4 VOLT
       CAPABILITY CURRENT=1 AMP
  CNX P=HI, Q=LO;
```



Thus:

a) Each resource component has two pins; P and Q.

b) Components 1 through 2 of set A (subset B) have voltage=4 volt as globally defined.

c) Components 3 through 4 of set A (subset C) have voltage=1 volt as locally defined at the C level.

d) Components 5 through 7 of set A (subset D) have a voltage=3 volt specification, however only the components 5 through 6 of set A (1 through 2 of subset D, subset E) have that specification since component 7 of set A (component 2 of subset D, subset F) has the voltage respecified as voltage=2 volt.

e) Each resource component has a current capability of 1 amp.

The after-clause serves to define a time delay between the 'change' command and the actual change being invoked. The example shows a definition and usage with the CHANGE statement.

```
REQUIRE DSSD DC_LOGIC SOURCE SET
            (DATA(10), CLK AFTER DELAY 10 N*SEC TO 100 N*SEC)
            WITH -------------
            CNX --------------;


CHANGE STATE OF DSSD.DATA TO SV.DATA,
                DSSD.CLK TO BIN '1'
                        AFTER_DELAY = 50 N*SEC ;
```

When the CHANGE statement is executed, the above statements cause resource set DSSD.DATA to change value to SV.DATA at the time of 'change' command (or t-zero). The resource DSSD.CLK is changed to the value of BIN '1' after a delay of 50 nanoseconds from the time of the 'change' command (or t-zero+50).

III. RESOURCE REFERENCES

Resources are referenced by enumeration of the resource name.

1. Syntax

resource-ref ::= resource-name . ... [:component-selector:]

component-selector ::= integer-constant

2. Constraints

If the resource name represents a set, the name represents the entire set. If it is desired to reference a subset, it is required to enumerate the base set and the subset. If it is required to reference an individual component of a set or subset, it is necessary to index into the set or subset base.

3. Examples

Given the following resource REQUIREment

```
REQUIRE JAK DC_LOGIC SOURCE SET
          (A(6), B(C(2), D(2)), 4) -----
```

JAK	Entire set of 14 components.
JAK.A	Subset A which is equivalent to components 1 through 6 of the base set (JAK).
JAK.B	Subset B which is equivalent to components 7 through 10 of the base set (JAK).

JAK.B.C	Subset C of subset B of base set JAK which is equivalent to components 1 through 2 of subset B which are equivalent to components 7 through 8 of the base set JAK.
JAK.B.D:1:	The first component of the subset D which is equivalent to the third component of the subset B which is equivalent to the ninth component of the base set JAK.

It can be seen from the above examples that any set, sub-set or single component of a set can be referenced either by a set-name,subset name or by a selector clause or a combination of both techniques.

4. *Semantics*

If the subsets are referenced, they must be based to the next highest level structure (set or subset). This is continued recursively until the base set is encountered.

The integer-constant of the selector clause represents the ordinal index within the set.

N.B. The left most resource component in the set is represented as ordinal position 1.

IV. DECLARE STATEMENT

To aid in examining and setting substrings, the following nomenclature changes are recommended.

1. Syntax (Changes)

```
string-mode ::= { CHARSTRING } [ ( { length-clause,... } ) ]
                { BITSTRING   }   ( {       *        } )

length-clause ::= { integer-constant                      }
                  { name [(length-clause,... )]           }
```

2. Examples

```
1)  DECLARE S BITSTRING(80)  $
2)  DECLARE S BITSTRING(L(40),R(80))  $
3)  DECLARE X CHARSTRING(6,SZ(10),4,Q(8,QZ(1),2))  $
```

Examples 1 and 2 both declare a bitstring identifier S of 80 bits. However, in example 2, the string is constructed from the concatenation of the substrings L and R, each consisting of 40 bits. Example 3 shows a more complex structured example.

V. STRING REFERENCES

Strings may be referenced in their entirety by enumeration of the base identifier.

1. Syntax

```
string_ref  ::=  name,...[:string_selector:] [(array_selector)]

string-selector  ::=  expression

array_selector  ::=  expression,...
```

2. Constraints

If it is desired to reference a substring of bits (or characters), it is required to enumerate the base string identifier and the substring identifier, or by an index reference, or by a combination of both techniques.

3. Examples

Given the following declare statements:

```
1)  DECLARE X CHARSTRING(6,SZ(10),4,Q(8,QZ(1),2))  $
2)  DECLARE S BITSTRING (L(40),R(40))  ARRAY (100)  $
```

From 1

```
X.SZ  means  Characters 7 thru 16 of the string X.
X.Q.QZ  and  X.Q:9:  and  X:29:  are equivalent
                    and mean character 9 of sub-string X.Q
                    or character 29 of the string X.
```

From 2

```
S.L(7)  means bits 1 thru 40 of element 7 of array S.
S.R(7)  means bits 41 thru 80 of element 7 of array S.
S.R  means the sub-string array consisting of bits 41 thru 80
```

of the array S.

4. Semantics

If the substrings are referenced, they must be based to the next highest level structure (string or substring). This is continued recursively until the base set is encountered.

The expression of the string-selector clause represents the ordinal bit or character index within the component structure.

N.B. The left-most character or bit is considered as ordinal position 1.

VI. SPECIFY STATEMENT

A SPECIFY statement describes an actual test-point on a UUT, and the constraints, if any, to be imposed on that point.

1. Syntax

```
specify-stmt   ::= SPECIFY test-place,... { linkage restrictions ;
                                          { shape

linkage        ::= LINKED connection,...

connection     ::= pin-descriptor = test-point

pin-descriptor ::= name

test-point     ::= name

shape          ::= SET (shape-clause)

shape-clause   ::= { integer-constant                    } [restrictions]
                   { test-point [(shape-clause,...)]     }

restrictions   ::= AS { { resource-name } } { EMITTER  } { specify-spec }
                      { { noun  modifier } } { ACCEPTOR }

                      ORAS...

specify-spec   ::= WITH limitation,...

limitation     ::= modifier [modifier-value],...
```

```
modifier-value  ::= { { <  }
                    { { <= } numeric-constant
                    { { >  }
                    { { >= }
                    {  =          numeric-constant TO numeric-constant
                    {  =          numeric-constant
```

2. Constraints

If a resource-name is part of a SPECIFY statement, it shall have been previously described in a REQUIRE statement. Pin-descriptors and modifiers must be appropriate to the indicated noun. Test-point names at the same level within a shape-clause must be distinct.

3. Examples

1) SPECIFY UUT SET(100) $

2)
```
SPECIFY UUT SET
        (DIGITAL(CLOCK(4),DATA(16),CONTROL(8))
                LOGIC VOLTAGE ACCEPTOR WITH
                      VOLTAGE_0<2.2 VOLT,
                      VOLTAGE_1>=2.2 VOLT,
         ANALOG(2)
                DC VOLTAGE EMITTER WITH
                  VOLTAGE=0 VOLT TO 10 VOLT);
```

4. Semantics

In Example 1, a UUT is modelled as a set of 100 ordered test points any of which may be referenced by subscripting.

In Example 2, the UUT is structured as a set of 30 test points which are grouped into two first level subsets consisting of DIGITAL (28 test-points) and ANALOG (2 test-points). The ANALOG subset is restricted to a DC voltage emitter with a voltage range of 0 through 10 volts. The DIGITAL subset is restricted to a logic voltage acceptor with a voltage_0 represented by less than 2.2 volts and voltage_1

represented by greater than or equal to 2.2 volts. The subset DIGITAL is further subset into three groups; CLOCK, DATA and CONTROL, with 4, 16, and 8 test-points, respectively.

VII. TEST POINT REFERENCE

Test points are referenced by enumeration of the test-point name.

1. Syntax

test-point-ref ::= test-point. ...[:test-point-selector:]

test-point-selector ::= integer-constant

2. Constraints

If the test-point name represents a set, the name represents the entire set. If it is desired to reference a sub-set, it is required to enumerate the base set and the sub-set. If it is required to reference on an individual pin of a set or a sub-set, it is achieved by indexing either to a set or sub-set.

3. Examples

Given Example 2, under SPECIFY above,

UUT	Entire set of 30 pins
UUT.DIGITAL	Sub-set DIGITAL which is equivalent to pins 1 through 28 of the base set UUT.
UUT.DIGITAL.DATA	Sub-set DATA of sub-set DIGITAL of set UUT equivalent to pins 5 through 20 of sub-set DIGITAL and pins 5 through 20 of set UUT.
UUT.DIGITAL.CLOCK:2:	Pin 2 of sub-set CLOCK.

4. Semantics

If sub-sets are referenced, they must be based to the next highest level structure (set or sub-set). This is continued recursively until the base set is encountered.

The integer-constant of the selector clause represents the ordinal index within the set.

N.B. The left most pin in the set is considered as ordinal position 1.

VIII. CONNECTION REFERENCES

The following construct permits the reference of sets of resource pins and connecting them to sets of test points.

1. Syntax

source-conn ::= {resource-pin = test-point-group,...},...

test-point-group ::= test-point-ref
(test-point-ref,...)

test-point-ref ::= name. ... [:test-point-selector:]

test-point-selector ::= integer-constant

sensor-conn ::= { resource-pin = { test-point-ref
(test-point-ref,...) } ,... }

2. Constraints

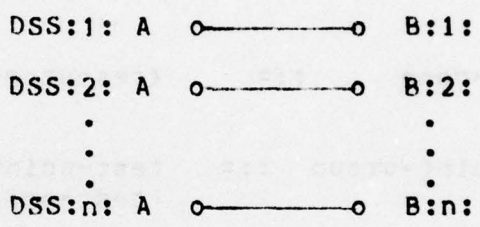
If parenthetical phrases are written to the right of the equals sign, the resource pin references to the left of the equals must be a set. Further, the components of the resource pin set are connected to the individual components described in the parenthetical phrase in the order in which they appear. If several test-point-group constructs are written with intervening commas, each component of the resource pin set is connected to the corresponding component in each test-point-group.

3. Examples

1) CONNECT DSS AT A=B;

The resource-pin (set) A is connected to the test-point (set) B. If A is a set(associated with

the resource DSS) then B must be a set of the same size. The connection would be made such that the pin A of resource DSS:I: would be connected to test-point B:I: for 1<=I<=n , where n is the number of components in the resource set A and the number of test-points in the pin set B. This statement corresponds to the following diagram:



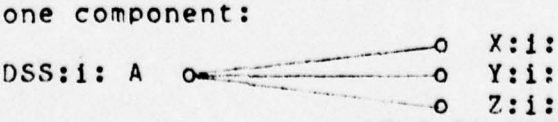
2) CONNECT DSS AT A=(B,C.D);

The components of resource-pin set A are connected to the components of test-point set B and sub-set C.D in order. If A is of size p, B is of size q, and C.D is of size r, then q+r must equal p. Further, the first component of A is connected to the first component of B and the (q+1)th component of A is connected to the first component of C.D.

3) CONNECT DSS AT A=X,Y,Z;

The resource-pin (set) A is fanned out to all of X,Y, and Z. If A,X,Y, and Z are single component entities then the meaning of the statement is that A is fanned out to X,Y, and Z. If A,X,Y, and Z are sets, they must all be of the same size. Further, each component of A is fanned out to the corresponding components of X, Y, and Z.

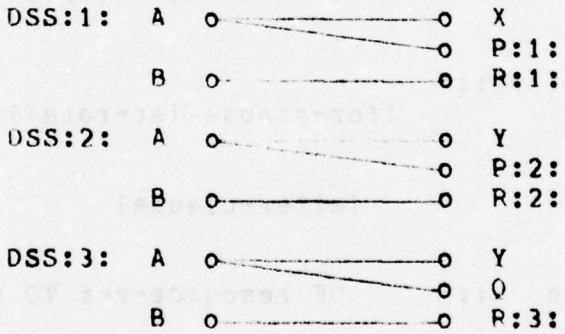
For the case where A,X,Y, and Z are sets, the following diagram describes one component:



4) CONNECT DSS AT A=(X,Y,Z),(P,Q),B=R

If resource DSS has been described as a 3 component set with each component having resource pins A and B, this statement directs that each component of A is fanned out to two places and each component of B is connected to the corresponding component of P. The first component of A is connected to both X and the first component of P (assuming P is a set of size 2 and Q is a single point). The third component of A is connected to both Z and Q.

This statement corresponds to the following diagram:



4. Semantics

The source-conn (and sensor-conn) construction describes the connection of resource pins to test-points. It permits the description of resource pins fanned out to a number of test points. It further allows the description of the connection of sets of resource pins to individual test points or sets of test points.

N.B. When resource sets are described in the REQUIRE statement there is an implicit definition of one or more connection sets. In the following statement:

```
REQUIRE A DC VOLTAGE SOURCE SET(20)  CNX X Y ;
```

we have implicitly declared connection sets X and Y each having 20 pins. The need to reference them without the equivalent resource does not arise but the programmer must be aware that they exist.

IX. CHANGE STATEMENT

This statement is used to cause a specified resource characteristic to be changed.

1. Syntax

change-stmt ::= [label:] change-group ;

change-group ::= CHANGE noun modifier change-structure,...

change-structure ::= [for-clause [at-rate]] { change-clause | (change-structure,...) }

[after-clause]

change-clause ::= OF resource-ref TO value [for-time]

at-rate ::= AT_RATE rate-clause

rate-clause ::= { { gt-lt | eq } rate-value | rate-value TO rate-value }

rate-value ::= expression

after-clause ::= AFTER_DELAY time-clause

for-time ::= FOR_TIME time-clause

time-clause ::= { { gt-lt | eq } time-interval | time-interval TO time-interval }

resource-ref ::= name. ...[:component-selector:]

component-selector ::= integer-constant

2. Constraints

The named modifier must have been associated with the resource in a REQUIRE statement. If more than a single resource is referenced in the iterative or recursive structures, they must all be components of the same top-level set of resources. If the at-rate clause is utilized , the rate modifier must have appeared as a modifier in the REQUIRE statement. If the after-clause is utilized, a corresponding after-clause must have appeared in the REQUIRE statement. If the for-time clause is utilized, it must have also appeared as a modifier in the REQUIRE statement.

3. Examples

```
1) CHANGE AC VOLTAGE OF ACS TO 8 VOLT;
2) CHANGE DC VOLTAGE OF DCS1 TO 4 VOLT FOR_TIME=2 SEC;
3) CHANGE DC VOLTAGE (OF DCS1 TO 4 VOLT,
                      OF DCS1 TO 0 VOLT AFTER_DELAY=2 SEC);
4) CHANGE DC VOLTAGE FOR I=1 TO 8 AT RATE=10 K*HZ
                      (OF DCS1 TO 4 VOLT FOR_TIME=1 U*SEC);
```

An analysis of the examples follows:

1) Assuming the current value of ACS were 0 volt, it is changed to 8 volts for an indefinite period.

2) Assuming the current value of DCS were 0 volt, it is changed to a value of 4 volts for a period of 2 seconds and then returns to the value of 0 volt. It should be understood that the action following that statement will occur concurrently with the change.

3) Assuming the current value of DCS were 0 volt, it is changed to a value of 4 volts and program execution is suspended for a period of 2 seconds, after which the value is returned to 0 volt. Unlike example 2 the action following the statement does not

occur until after the value of DCS has returned to 0 volt.

4) Assuming the current value of DCS were 0 volt, this statement produces a burst of 8 pulses of amplitude 4 volts, a pulse-width of 1 usec, and a pulse-repetition rate of 10 KHZ.

4. Semantics

The value of the changed characteristic is changed as specified by the change-clause(s). If the for-time clause is utilized, the change occurs for the duration of the specified time after which the characteristic returns to the original value. It should be noted that the action following such usage occurs concurrently, or as nearly so as the system will allow. If the after-clause is utilized, it refers to the preceding clause and implies a delay from the CHANGE command or, if the resource reference is a sub-set then the delay is from the CHANGE of the next outermost set (or subset). If the for-clause is utilized, the following group is iterated as specified at either a rate controlled by the rate-clause or of an unspecified rate. If unspecified, the rate is assumed to be as fast as the system will allow.

X. READ Statement

A READ statement causes a single or multiple read of UUT characteristics and the storage of the results.

1. Syntax

```
read-stmt        ::=   [label:] read-group ;

read-group       ::=   READ[noun  real-modifier IN unit[accuracy]
                                  string-modifier
                       read-structure

read-structure   ::=   [for-clause [at-rate]]  read-clause
                                               (read-structure,...)
                       [after-clause]

read-clause      ::=   USING resource-ref INTO location

at-rate          ::=   AT-RATE rate-clause

rate-clause      ::=   gt-lt  rate-value
                       eq
                       rate-value TO rate-value

after-clause     ::=   AFTER_DELAY  gt-lt  time-interval
                                    eq
                                    time-interval TO time-interval

rate-value       ::=   expression

time-interval    ::=   expression
```

2. Constraints

The named noun and modifier, if used, must be the same as those in the REQUIRE statement. If more than a single resource is referenced in the iterative or recursive structure, they must all be components of the same base set. If the at-rate clause is utilized, the data-rate modifier must have appeared as a modifier in the REQUIRE statement. If the after-clause is utilized, that clause must have appeared within the resource set definition.

3. Examples

```
1)      READ AC VOLTAGE IN M*VOLT USING MVA-1 INTO VI ;


2)      READ DC VOLTAGE IN VOLT (USING DVM INTO V1,
                                 USING DVM INTO V2)
                                   AFTER-DELAY = 100 M*SEC ;


3)      READ DC VOLTAGE IN VOLT
                FOR I=1 TO 10 AT_RATE = 1 K*HZ
                        (USING DVM INTO V(I))  ;


4)      READ DC_LOGIC STATE
                FOR I=1 TO 8 AT_RATE = 10 K*HZ
                        USING DSSR.S INTO RW:I:  ;
```

Explanations

1) Read the value measured by the resource MVA-1; scale the value into volts, if necessary, and store the result in location VI.

2) Read the value measured by the resource MVA-1; scale the value into volts, if necessary, and store the result in location V1. After a delay of 100 milliseconds repeat the operations but store the result in V2.

3) Read the value measured by the resource DVM; scale the value into volts, if necessary, and store the result in the elements (1 through 10) of the array V. The 10 readings shall be made at a rate of 1 kilohertz.

4) The digital logic sensor DSSR.S will read an 8-bit serial data item at a bit rate of 10 kilohertz and store the result into the bit-string data word RW such that the first bit received becomes bit 1 of the word, (i.e., most significant bit first).

4. Semantics

The current value(s) being sensed by the named resource is recorded, scaled into the specified units, if necessary, and stored in the specified location.

XI. RUN Statement

The RUN statement provides a means of concatenating a simple or compound change-structure with a simple or compound read structure.

1. Syntax

```
run-stmt          ::=   [label:] RUN run-structure ;

run-structure     ::=   [for-clause,... [at-rate]]
                            change-read-group

change-read-group ::=   (change-group) (read-group)

for-clause        ::=   { FOR variable = list
                        { FOR variable = iterator,...
                        { FOR expression

variable          ::=   name

list              ::=   expression

iterator          ::=   expression [BY expression] TO expression

change-group      ::=     CHANGE noun modifier change-structure,...

change-structure  ::=
                        [for-clause [at-rate]]  { change-clause
                                                { (change-structure,...)
                            [after-clause]

change-clause     ::=     OF resource-ref TO value [for-time]
```

```
resource-ref      ::=   name. ...[:component-selector:]

component-selector ::=  integer-constant

read-group        ::=   READ[noun {real-modifier IN unit[accuracy]}
                                  {string-modifier                }
                        read-structure,...

read-structure    ::=   [for-clause [at-rate]]  {read-clause          }
                                                {(read-structure,...)}
                        [after-clause]

read-clause       ::=   USING resource-ref INTO location

at-rate           ::=   AT-RATE rate-clause

rate-clause       ::=   {{gt-lt}     rate-value     }
                        {{eq  }                    }
                        {rate-value TO rate-value }

after-clause      ::=   AFTER_DELAY {{gt-lt} time-interval     }
                                    {{eq  }                  }
                                    {time-interval TO time-interval}

rate-value        ::=   expression

time-interval     ::=   expression
```

2. Constraints

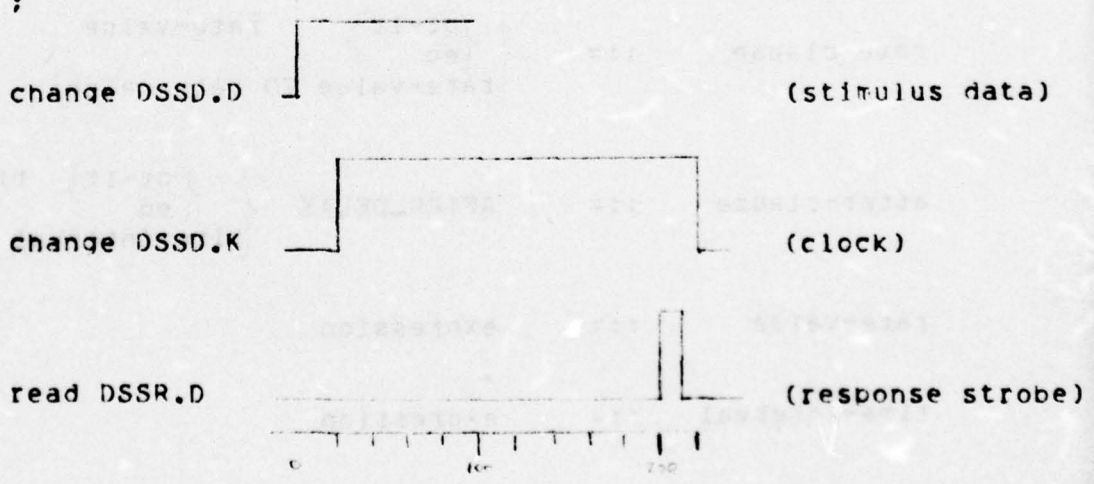
If the initial and final expression of a for-clause is iterated, the implication is that the data may be segmented for each iteration group. The data rate for each segment is guaranteed to be that specified, but an indeterminate time

may exist between the transmission of segments. This facility permits the programmer who has knowledge of the UUT to show how the data may be segmented by the compiler. If such iterations are employed, the BUFFER_SIZE modifier must have been used with the resource REQUIRE statement.

3. Examples

1)
```
RUN
        (CHANGE DC VOLTAGE OF DCS TO 10 VOLT)
        (READ DC VOLTAGE IN VOLT
                FOR I=1 TO 10 AT_RATE=100 HZ
                        (USING DVM INTO V(I)))   ;
```

2)
```
RUN FOR I=1 TO 100 AT_RATE = 1 M*HZ
        (CHANGE AC_LOGIC STATE (OF DSSD.D TO SV(I),
                                OF DSSD.K TO HI
                                    FOR 100 N*SEC
                                    AFTER_DELAY=20) N*SEC
        )
        (READ DC_LOGIC STATE USING DSSR.D INTO RV(I)
                                AFTER_DELAY 200 N*SEC
        )
    ;
```



Analysis of Examples

1) Assuming the resource DCS had been applied at 0 volts, a transient response is being read. The forcing function is a 10 volt edge and 10 samples are being obtained at intervals of 10 milliseconds and stored in elements 1 through 10 of the array V. The first sample is taken concurrently with the forcing function transition.

2) A simple NRZ parallel digital test is being performed. There are 100 stimulus patterns obtained from the bitstring array SV. The stimulus logic type is on AC signal, i.e., probably an fsk signal. To ensure settling of the stimulus drivers, the clock (DSS,K) is delayed by 20 nanoseconds. The clock pulse width is 100 nanoseconds. The sensor (DSSR,D) is DC logic and reads the response pattern after a delay of 200 nanoseconds from the issuance of the stimulus pattern. The stimulus-clock-response sequence is reiterated with a period of 1 microsecond.

4. Semantics

The RUN statement is the vehicle to link a change group referencing a source resource (set) to a read group referencing a sensor resource (set). It permits precise time control between the change and read actions. In addition, a single level iterative capability is provided.

Attachment I
Change Pages to OPAL Specification

The following change pages are keyed to the paragraph or appendix number found in the OPAL Specification. In general, they represent partial changes to the paragraphs identified.

Appendix A - Examples Using the New OPAL Digital Test Statements.

Example I - Parallel Data to FSK Signal Converter.

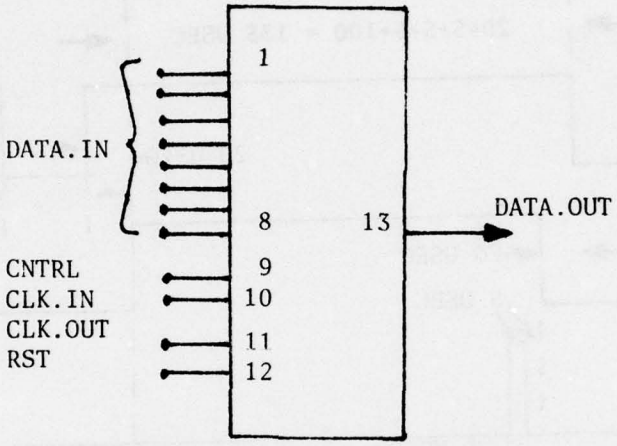


Figure 1 - Parallel Data to FSK Signal Converter

The device accepts an 8-bit parrallel dc-voltage logic input on pins 1 through 8. Pin 9 is a control function and should be held in a '1' state during all data transmission. Pin 10 is the input clock line and requires a change from '0' to a '1' state to capture the data on the input lines. Pin 11 is the output clock line and must complete 10 cycles at a rate of 100 KHZ to strobe the data out in a serial fashion. Ten cycles are required since the device supplies a single start bit and a single stop bit. Pin 12 is a reset line and must be pulsed from a '0' state to initialize the controller. Pin 13 is the ac-frequency serial output.

Pins 1 through 12 carry dc-logic where a logic '0' is represented by 0 volts, and logic '1' by 4.7 volts. The threshold is 2.4 volts. At the output, Pin 13 outputs an ac-logic signal where a logic '0' is represented by a frequency of 0.98 MHZ and a logic '1' by 1.02 MHZ. The threshold is considered to be 1.00 MHZ. In either logic state the voltage level is approximately 4.2 volts rms.

Timing Requirements

The following diagram, Figure 2, details the required timing constraints for each frame of input data.

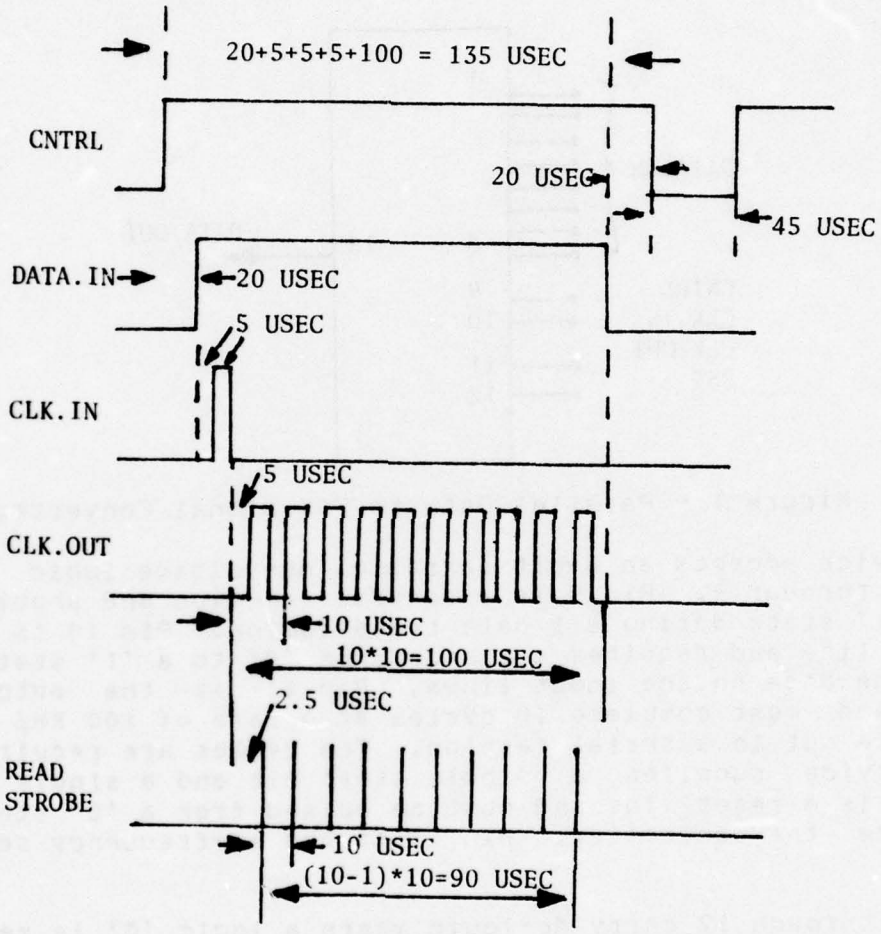


Figure 2 - Timing Diagram

Each frame consists of a total active period of 155usec. Allowing a relaxation period of 45usec will result in a total period of 200usec, thus, sequential tests will be run at a rate of 5 KHZ.

Definitive Statements

From the information contained in the description and timing requirements, the following OPAL statements can be derived:

UUT Specification:

```
SPECIFY UUT SET
          (IN(DATA(8),CNTRL,CLK(I,O),RST)
              AS DC_LOGIC VOLTAGE ACCEPTOR WITH
              VOLTAGE_0 < 2.4 VOLT,
              VOLTAGE-1 >= 2.4 VOLT
           OUT AS AC_LOGIC FREQUENCY EMITTER WITH
              VOLTAGE = 4.7 VOLT,
              FREQUENCY_0 = 0.98 M*HZ,
              FREQUENCY_1 = 1.02 M*HZ
          ) ;
```

Stimulus Requirements:

```
REQUIRE DIGSTIM DC_LOGIC VOLTAGE SOURCE SET
                (CNTRL      WITH CAPABILITY
                                 DATA_WIDTH = 20+5+5+5+100+20 U*SEC,
                                 DATA_RATE = 5 K*HZ
                 DATA(8)    AFTER_DELAY = 20 U*SEC
                            WITH CAPABILITY
                            DATA_WIDTH = 5+5+5+100 U*SEC,
                            DATA_RATE = 5 K*HZ,
                            BUFFER_SIZE = 256,
                 CLOCK( IN WITH CAPABILITY
                                 DATA_RATE = 5 K*HZ,
                       OUT AFTER_DELAY = 5 +5 U*SEC
                            WITH CAPABILITY
                                 DATA_RATE = 100 K*HZ
                      ) AFTER_DELAY = 20+5 U*SEC
                        WITH CAPABILITY
                             DATA_WIDTH = 5 U*SEC
                 RST    WITH CAPABILITY
                             DATA_WIDTH = 10 U*SEC
                )  WITH CAPABILITY
                       VOLTAGE_0 = 0 VOLT,
                       VOLTAGE_1 = 4.7 VOLT
                   CNX S = TRU ;
```

NOTE: Notice the recursive definition of the DIGSTIM resource set and the semantics of the after-clause. The subset DIGSTIM.CLOCK is delayed, with respect to the set DIGSTIM, by 25 usec (20+5). The subset DIGSTIM.CLOCK.OUT is delayed, with respect to the subset DIGSTIM.CLOCK, by 10 usec (5+5), thus, the total delay of DIGSTIM.CLOCK.OUT with respect to the set DIGSTIM, is 35 usec (25+10) in accordance with the timing diagram.

Response Requirements:

```
REQUIRE DIGRESP AC_LOGIC FREQUENCY SENSOR
                         WITH CAPABILITY
                         FREQUENCY_0 < 1 M*HZ,
                         FREQUENCY_1 >= 1 M*HZ
                            LIMIT
                         VOLTAGE > 5 VOLT
                         CNX R = HI;
```

The stimulus data array is defined as follows

```
        DECLARE STIM_DATA BITSTRING(8) ARRAY(256);
```

In the test that follows, it is assumed that the stimulus data has been appropriately initialized.

The response data array is defined as follows:

```
    DECLARE RESP BITSTRING(STB,DATA(8),SPB) ARRAY(256);
```

Note that the substrings STB and SPB represent the start and stop bit, respectively.

```
     //CONNECT AND INITIALIZE STIMULUS SUB_SYSTEM TO UUT
INIT:  APPLY DIGSTIM WITH
                     VOLTAGE_0 = 0 VOLT,
                     VOLTAGE_1 = 4.7 VOLT,
                     STATE = HEX '0000'
                     AT S = (UUT.IN.CNTRL,
                             UUT.IN.DATA,
                             UUT.IN.CLK,
                             UUT.IN.RST) ;
     //APPLY RESET PULSE OF 10 U*SEC TO UUT
RESET:  CHANGE DC_LOGIC STATE OF DIGSTIM.RST TO BIN '1' FOR 10 U*SEC;

     //CONNECT AND INITIALIZE SENSOR SUB_SYSTEM TO UUT.
       INITIATE DIGRESP  WITH
                         FREQUENCY_0 < 1.0 M*HZ,
                         FREQUENCY_1 >= 1.0 M*HZ,
                         VOLTAGE     <= 5.0 VOLT
                         AT
                         R = UUT.OUT ;

     //NOW WE ARE READY TO TEST

     //ALL 256 PATTERNS WILL BE OUTPUT CONTIGUOUSLY WITHIN
     //THE TIME FRAMES SPECIFIED IN FIG.2.

TEST:  RUN FOR I = 1 TO 256 AT_RATE = 5 K*HZ
```

```
(CHANGE DC_LOGIC STATE
        (OF DIGSTIM.CNTRL TO BIN '1' FOR_TIME = 155 U*SEC,
         OF DIGSTIM.DATA TO STIM_DATA(I) FOR_TIME = 135 U*SEC
                                AFTER_DELAY = 20 U*SEC,
        (OF DIGSTIM.CLOCK.IN TO BIN '1' FOR_TIME = 5 U*SEC,
        (FOR J=1 TO 10 AT_RATE = 100 K*HZ
           OF DIGSTIM.CLOCK.OUT TO BIN '1' FOR_TIME = 5 U*SEC
        )                               AFTER_DELAY = 10 U*SEC
        ) AFTER_DELAY = 25 U*SEC
        )
)
        (READ AC_LOGIC STATE
             (FOR K = 1 TO 10 AT_RATE = 100 K*HZ
                USING DIGRESP INTO RESP:K:(I)
             ) AFTER_DELAY = 37.5 U*SEC
        ) ;
```

Example 2 - Digital to Analog Converter

The device is a digital to analog converter. The digital input is on lines 1 through 10 where line 1 represents the most significant bit. The logic is dc-voltage where 1.8 volts represents a logic '1' and -3.6 volts represents a logic '0'. The analog output covers the range 0 to 10.27 volts dc, thus, the least significant bit represents 0.01 volts. Line 11 is a strobe and must be held in a '1' state to cause the digital data to be captured. The response time from the application of the strobe to settling of the analog output is 100 usec. In addition the strobe should be delayed, with respect to the data, by 10 usec for reliable data capture.

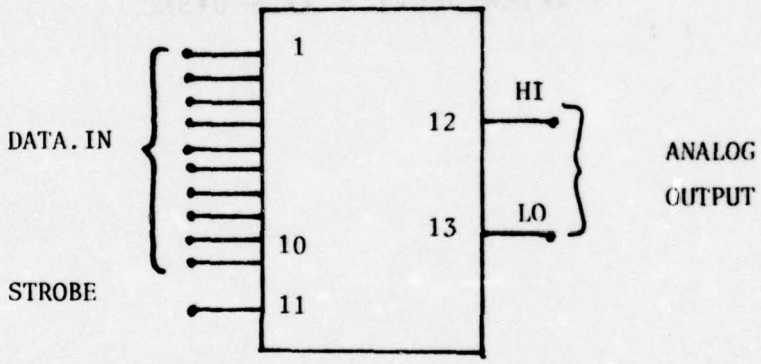


Figure 1 - Digital to Analog Converter

Figure 2 shows the timing relationships between the signals.

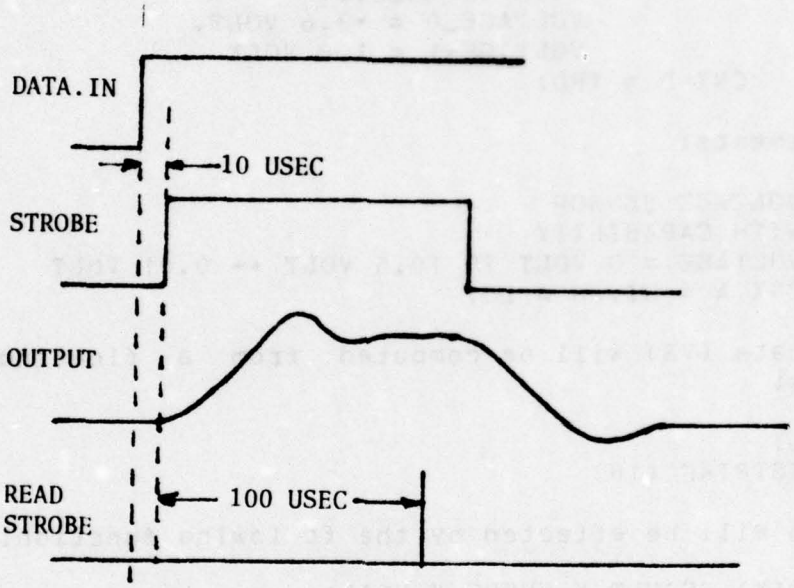


Figure 2 -Timing Requirements

Definitive Statements

From the information contained in the description and timing requirements the following OPAL statements can be derived.

UUT Specification:

```
SPECIFY  D_A SET
            (DIGITAL(DATA(10),STROBE)
                  AS DC_LOGIC VOLTAGE ACCEPTOR WITH
                  VOLTAGE_0 > -0.9 VOLT,
                  VOLTAGE_1 <= -0.9 VOLT
             ANALOG (2)
                  AS DC VOLTAGE EMITTER WITH
                  VOLTAGE = 0 VOLT TO 10.5 VOLT
```

```
        ) ;

Digital Requirements:

REQUIRE DIGSTIM DC_LOGIC VOLTAGE SOURCE SET
                (DATA(10) AFTER_DELAY = 10 U*SEC,
                 STROBE)
                          WITH CAPABILITY
                          VOLTAGE_0 = -3.6 VOLT,
                          VOLTAGE-1 = 1.8 VOLT
                 CNX D = TRU;

Analog Requirements:

REQUIRE A DC VOLTAGE SENSOR
             WITH CAPABILITY
             VOLTAGE = 0 VOLT TO 10.5 VOLT +- 0.01 VOLT
             CNX A = HI, B = LO;

The stimulus data (VS) will be computed  from  a  floating  point
value (V) thus:

DECLARE V REAL;
DECLARE VS BITSTRING (10)

The conversion will be effected by the following function:

DEFINE BIT_FLT(X) RESULT Y WHERE X REAL;
DECLARE Y BITSTRING(10);
DECLARE I, M INTEGER;
SET M = FIX(X)
REPEAT FOR I=1 TO 10;
IF REM (M,2) = 1 THEN
   SET Y:10-I+1:=BIN '1';
END IF;
SET M = INTDIV (M,2);
END REPEAT;
RETURN;
END DEFINE BIT_FLT;
     // CONNECT AND INITIALIZE STIMULUS SUB_SYSTEM TO UUT.

INIT:  APPLY DIGSTIM WITH
             VOLTAGE_0 = -3.6 VOLT,
             VOLTAGE_1 = 1.8 VOLT,
             STATE = BIN '00000000000'
             AT D = D_A.DIGITAL;
       // CONNECT AND INITIALIZE SENSOR SUB_SYSTEM TO UUT
```

```
      INITIATE A WITH
                  VOLTAGE = 0 VOLT TO 10.5 VOLT
                  AT A = D_A.ANALOG:1:, B=D_A.ANALOG:2:;

      // NOW LETS GO

      REPEAT FOR V = 0 BY 0.01 TO 10.27;
             SET VS = BIT_FLT(100*V);
             RUN
                (CHANGE DC_LOGIC STATE
                        (OF DIGSTIM.DATA TO VS,
                         OF DIGSTIM.STROBE TO BIN '1' AFTER 10 U*SEC
                        )
                )
                ( READ DC VOLTAGE IN VOLT
                                  USING A INTO VM AFTER 110 U*SEC
                );
             CHANGE DC_LOGIC STATE OF DIGSTIM.STROBE TO BIN '0';
             IF ABS(VALUE_OF(VM)-V) > 0.01 THEN GO_TO ERR;
             END IF ;
  END REPEAT ;

  // IF CONTROL ARRIVES HERE ALL IS WELL.
     . . . .
     . . . .
  // IF CONTROL ARRIVES HERE THE TEST DID FAIL.
ERR: . . .
```

Example 3 - Fault Isolation

The following program assumes a UUT with 32 input pins and 64 output pins. The actual digital test data is stored on an external device in the following format:

```
nnnnn      (number of test patterns)
nnnnn      (number of possible faults)
nnnnn      (number of fault patterns in dictionary)
bbbbbbbb(32 bits of 0,1, or X) bbbbb   input pattern
bbbbbbbb (64 bits of 0,1, or X) bbbbb   output pattern
   .                              .
   .                              .
   .                              .
--- the above pair repeats for the indicated number of tests ---
.....

ttttt nnnnn bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
  .     .              .
  .     .              .
  .     .              .
---- the above pattern  test number, fault number, and fault pattern
---- repeats for the indicated number of fault patterns
---- the bit pattern for the fault dictionary is also in
---- 0,1, and X with the interpretation:

                0 = no error expected
                1 = error expected
                X = ignore this pin
```

The basic algorithm is the following:

1. Apply each stimulus and read the response

2. If response differs from the expected "good" response then save the error vector on the intermediate tape.

3. At end of test if there were any errors then

4. Clear fault table to 0's.

5. Read the faults off the input dictionary and

6. For each bit in the output compare and take the action

7. dict bit | act resp bit | modify weight

dict bit	act resp bit	modify weight
0	X	0
1	0	+3
1	1	+1

8. At end locate list of 20 highest valued faults and print

```
BEGIN OPAL PROGRAM DIGTEST;

   DECLARE IPAT CHARSTRING (64),  // INPUT STRING FOR CONVERSION
           STIMVEC BITSTRING (32),  // STIMULUS VECTOR
           RESPVEC BITSTRING (64),  // RESPONSE VECTOR
           DONTCARE BITSTRING (64),  // DONT CARES IN RESPONSE
           FAULTVEC BITSTRING (64),  // FAULT VECTOR
           ZEROVEC BITSTRING (64) INITIAL
BIN '0000000000000000000000000000000000000000000000000000000000000000'  ;

   DECLARE WEIGHT INTEGER ARRAY(4000);

   DECLARE I,    // LOCAL TEMPORARY
           J,    // ANOTHER LOCAL TEMPORARY
           LTFR,    // LAST TEMP FAULT READ
           TEST,    // CURRENT TEST NUMBER
           NPATS,    // NUMBER OF PATTERNS OF STIMULUS ON TAPE
           NERRVEC,    // NUMBER OF ERROR VECTORS ON TAPE
           LASTFLTTEST;    // LAST FAULT PATTERN ON INTERMEDIATE TAPE

   REQUIRE INTAPE DATA_BASE INPUT OPEN_TO_READ;//SOURCE OF INPUT
   REQUIRE PRINTER HARD_COPY OUTPUT OPEN_TO_WRITE;//FOR LISTINGS
   REQUIRE TEMPFILE TEST_RECORD INPUT_OUTPUT;//SCRATCH FILE


   REQUIRE STIM DC_LOGIC VOLTAGE SOURCE SE(32)
                WITH CAPABILITY VOLTAGE_1 = 4.7 VOLT,
                                VOLTAGE_0 = 0.3 VOLT
                CNX SO;

   REQUIRE RESP DC_LOGIC VOLTAGE SENSOR SET(64)
                WITH CAPABILITY VOLTAGE_1 > 2.5 VOLT,
                                VOLTAGE_0 <= 2.5 VOLT
                     LIMIT VOLTAGE < 5.0 VOLT;
                CNX SI;

   SPECIFY UUTO SET(64) AS DC_LOGIC VOLTAGE EMITTER;
```

```
SPECIFY UUTI SET(32) AS DC_LOGIC VOLTAGE ACCEPTOR;

//  ACTUAL PROGRAM STARTS HERE

TOP:

   SET LASTFLTTEST = 0;
   OUTPUT USING PRINTER FORMAT
        'INSERT UUT AND PRES GO!L(1)';


   DELAY UNTIL MANUAL_INTERVENTION;
   INPUT NPATS,NFAULTS,NERRVEC USING INTAPE FORMAT
      '#####!L(1)#####!L(1)#####!L(1)';
   IF NFAULTS>4000 THEN
      OUTPUT USING PRINTER FORMAT 'TOO MANY FAULTS!L(1)';
      DELAY UNTIL MANUAL_INTERVENTION;
      GOTO TOP;
   END IF;
   CONNECT STIM AT SO = UUTI;
   CONNECT RESP AT SI = UUTO;
   REPEAT FOR TEST = 1 TO NPATS;
      INPUT IPAT USING INTAPE FORMAT
        '################################!L(1)';
      SET STIMVEC = BIN(IPAT);
      INPUT IPAT USING INTAPE FORMAT
      SET RESPVEC = ZEROVEC;  SET DONTCARE = ZEROVEC;
      REPEAT FOR I = 1 TO 64;
         IF IPAT:I: = '1' THEN SET RESPVEC:I: =BIN'1'; END IF;
         IF IPAT:I: = 'X' THEN SET DONTCARE:I: =BIN'1'; END IF;
      END REPEAT;
      CHANGE LOGIC STATE OF STIM TO STIMVEC;
      READ LOGIC STATE USING RESP INTO RESPVEC;
      SET RESPVEC = RESPVEC AND NOT DONTCARE;
      IF RESPVEC NOT = ZEROVEC THEN
         OUTPUT TEST,RESPVEC
                USING TEMPFILE FORMAT '(5)# (64)#!L(1)';
```

(Note! Notice how much repetions inside of formats help!)

```
         SET LASTFLTTEST = TEST;
      END IF;
   END REPEAT;

// ALL OF THE INPUTS HAVE BEEN READ AND APPLIED.  WAS THE UNIT ANY GOOD?

   DISCONNECT STIM;  DISCONNECT RESP;
   IF LASTFLTTEST = 0 THEN    //UNIT WAS OK
```

```
      OUTPUT USING PRINTER FORMAT 'UNIT PASSED TEST!L(1)';
      GOTO TOP;
   END IF;
   OUTPUT USING PRINTER
                FORMAT 'UNIT FAILED, ANALYSIS STARTING!L(1)';
   REPEAT FOR I = 1 TO NFAULTS;
      SET WEIGHT[I] = 0;
   END REPEAT;
   OUTPUT USING TEMPFILE FORMAT REWIND;

   SET LTFR = 0;
   REPEAT FOR I = 1 TO NERRVEC;
      INPUT FAULT, IPAT USING INTAPE FORMAT '(5)# (64)#!L(1)';
      SET FAULTVEC = BIN(IPAT);
      IF I<=LASTFLTTEST THEN
         REPEAT WHILE LTFR<I ;
            INPUT LTFR, IPAT USING TEMPFILE
                        FORMAT '(5)# (64)#!L(1)';
            SET ERRVEC = BIN(IPAT);
         END REPEAT;
         ELSE SET ERRVEC = 0;
      END IF;

//  NOW SCAN ALL THE BITS

      REPEAT FOR J = 1 TO 64;
         IF FAULTVEC:I: =BIN '1' THEN
            IF ERRVEC:I: =  '1' THEN
               SET WEIGHT[FAULT] = WEIGHT[FAULT] + 1;
            ELSE
               SET WEIGHT[FAULT] = WEIGHT[FAULT] + 3;
            END IF;
         END IF;
      END REPEAT;
   END REPEAT;

   INPUT USING INTAPE FORMAT REWIND;
   OUTPUT USING TEMPFILE FORMAT REWIND;

// WEIGHT TABLE NOW BUILT, FIND 20 BIGGEST WEIGHTS

   SET BIGWEIGHTS[1] = 9999;
   REPEAT FOR I = 2 TO 21;
      SET BIGWEIGHTS[I] = 0;
      SET BIGFAULT[I] = 0;
   END REPEAT;
   REPEAT FOR I = 1 TO NFAULTS;
      REPEAT FOR J = 21 TO 1 WHILE WEIGHT[I]>BIGWEIGHT[J];
```

```
            SET BIGWEIGHT[J+1] = BIGWEIGHT[J];
            SET BIGFAULT[J+1] = BIGFAULT[J];
         END REPEAT;
         SET BIGWEIGHT[J+1] = WEIGHT[I];
         SET BIGFAULTS[J+1] = I;
      END REPEAT;

      OUTPUT USING PRINTER FORMAT
           'MOST PROBABLE FAULTS!L(2)FAULT  WEIGHT!L(2)';
      REPEAT FOR I = 2 TO 21;
         OUTPUT BIGFAULT[I], BIGWEIGHT[I] USING PRINTER FORMAT
           '#####  #####!L(1)';
      END REPEAT;
      GOTO TOP;

END OPAL PROGRAM DIGTEST;
```

5.2.4.1 DECLARE STATEMENT

To aid in examining and setting substrings, the following nomenclature changes are recommended.

5.2.4.1.1 Syntax (Changes)

```
string-mode ::= {CHARSTRING} [({length-clause,... })]
                {BITSTRING }  [({     *         })]

length-clause ::= {integer-constant                  }
                  {name [(length-clause,... )]       }
```

5.2.4.1.3 Examples

```
1)  DECLARE S BITSTRING(80)  $
2)  DECLARE S BITSTRING(L(40),R(80))  $
3)  DECLARE X CHARSTRING(6,SZ(10),4,Q(8,QZ(1),2))  $
```

Examples 1 and 2 both declare a bitstring identifier S of 80 bits. However, in example 2, the string is constructed from the concatenation of the substrings L and R, each consisting of 40 bits. Example 3 shows a more complex structured example.

5.2.4.3 REQUIRE Statement

A REQUIRE statement describes a virtual resource needed to conduct a test of a UUT and assumed to be available.

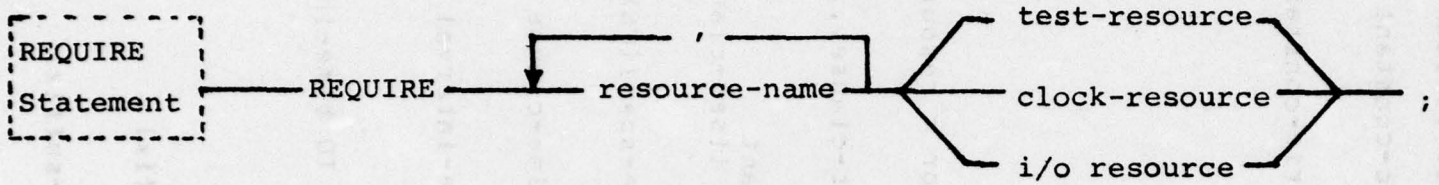
5.2.4.3.1. Syntax

```
require-stmt    ::=   REQUIRE resource-name,... { test-resource
                                                { clock-resource
                                                { i/o-resource   } ;

resource-name   :=    name

test-resource   ::=  { noun sig-modifier  {SOURCE}          }
                     {                    {SENSOR} } test-resource-desc
                     { IMPEDANCE sig-modifier LOAD          }

test-resource-desc  ::= [test-resource-structure] test-resource-spec

                        [CNX req-conn]

test-resource-spec  ::=  WITH [control-limitation][capability-limitation]

                         [limit-limitation]

control-limitation  ::= CONTROL req-limitation,...

capability-limitation  ::=  CAPABILITY req-limitation,...

limit-limitation    ::=  LIMIT req-limitation,...

req-limitation  ::=  { sig-limitation
                     { test-limitation
                     { thru-limitation }

sig-limitation  ::=   sig-modifier req-mod-desc

test-limitation ::=    test-modifier req-mod-desc

thru-limitation ::=    thru-modifier req-mod-desc

req-mod-desc    ::=  { req-mod-value
                     { req-mod-range }
```

```
req-mod-value   ::=   eq   { numerical-constant,...[accuracy]
                           { (numerical-constant accuracy),...

req-mod-range   ::=   {{gt-lt numeric-constant} |
                         {eq numeric-constant TO numeric-constant
                            [BY numeric-constant]}}[accuracy]

gt-lt           ::=   { >
                      { >=
                      { <
                      { <=

eq              ::=   =

req-conn        ::=   {pin-descriptor = resource-pin},...

test-resource-structure  ::=  SET (set-clause,...)

set-clause      ::=   { integer-constant                      }  [after-clause]
                      { resource-name [(set-clause,...)]      }
                         [test-resource-spec][CNX req-conn]

after-clause    ::=   AFTER_DELAY time-clause

time-clause     ::=   { { gt-lt }   time-interval              }
                      { { eq    }                             }
                      { time-interval TO time-interval        }

sig-modifier    ::=   name

noun            ::=   name[noun-suffix]

modifier        ::=   name[modifier-suffix]

noun-suffix     ::=   _name

modifier-suffix ::=   _name
```

5.2.4.3.2. Constraints

REQUIRE STATEMENT



Pin descriptors and modifiers must be appropriate to the indicated noun. Resource_pins and resource_names within a single set_clause and test_resource must be unique.

5.2.4.3.3. Examples

```
a)  REQUIRE T2 TIME INTERVAL CLOCK;

b)  REQUIRE ACS AC VOLTAGE SOURCE
            WITH CONTROL VOLTAGE=1 VOLT TO 10 VOLT BY 100 M*VOLT,
                         FREQ=50 HZ TO 400 HZ
                 CAPABILITY CURRENT=1 AMP
                 LIMIT CURRENT=2 AMP
            CNX HI=A, LO=B;

c)  REQUIRE ACVM AC VOLTAGE SENSOR
            WITH CAPABILITY VOLTAGE<=450 VOLT,
                            FREQ=50 HZ TO 440 HZ
            CNX HI=A, LO=B, LO=G;

d)  REQUIRE LD LOAD IMPEDANCE
            WITH CAPABILITY IMPEDANCE=1000 OHM
                 LIMIT      CURRENT_TRMS=1 AMP
            CNX P1=HI, P2=LO;

e)  REQUIRE DSSD DC_LOGIC SOURCE SET
                 (TTL(8) WITH CONTROL VOLTAGE_1=4.7 VOLT,
                                      VOLTAGE_0=0 VOLT
                               CAPABILITY CURRENT=100 M*AMP,
                  SLL(4) WITH CONTROL VOLTAGE_1=-1.2 VOLT ,
                                       VOLTAGE_0=5.8 VOLT
                 )  WITH LIMIT RISE_TIME < 100 N*SEC
                    CNX TRU=D, COMP=G;
```

5.4.3.3.4. Semantics

These semantics define only the structural concepts (i.e., Sets) for which this organization is responsible. The remainder of the statement semantics must be appended.

A single identifier (resource_name) may refer to a set of identical or similar resources. Any action referencing such an identifier implies that the action to each component of the set is to be performed concurrently. The number of components within the set is defined by the sum of the integer-constants which appear within the construct. Fully partitioned subsetting of contiguous groups of components, with or without names, may be specified by the recursive nature of the set-clause construct.

Test resource specifications (test_resource_spec) and connections (CNX req_conn) are local to the scope in which they are defined and global to any subsets within that structure with the following exception:

If they are respecified within a lower level structure, that respecification becomes valid for the scope in which it is defined. Of course this rule applies recursively.

The following example shows the scoping:

```
REQUIRE  A DC VOLTAGE SOURCE SET
          (B(2),C(2) WITH CONTROL VOLTAGE=1 VOLT,
           D(E(2),F WITH CONTROL VOLTAGE=2 VOLT)
             WITH CONTROL VOLTAGE=3 VOLT)
          ) WITH CONTROL VOLTAGE=4 VOLT
                CAPABILITY CURRENT=1 AMP
            CNX P=HI, Q=LO;
```

Component	Subset		Subset		Base	
1			1		1	
				B		
2			2		2	
3			1		3	
				C Voltage=		
4			2	1 Volt	4	A Voltage= 4 Volt
5	1		1		5	CNX P,Q
		E				
6	2		2	D Voltage= 3 Volt	6	
7	1	F Voltage= 2 Volt	3		7	

Thus:

a) Each resource component has two pins; P and Q.

b) Components 1 through 2 of set A (subset B) have voltage=4 volt as globally defined.

c) Components 3 through 4 of set A (subset C) have voltage=1 volt as locally defined at the C level.

d) Components 5 through 7 of set A (subset D) have a voltage=3 volt specification, however only the components 5 through 6 of set A (1 through 2 of subset D, subset E) have that specification since component 7 of set A (component 2 of subset D, subset F) has the voltage respecified as voltage=2 volt.

e) Each resource component has a current capability of 1 amp.

The after-clause serves to define a time delay between the 'change' command and the actual change being invoked. The

example shows a definition and usage with the CHANGE statement.

```
REQUIRE DSSD DC_LOGIC SOURCE SET
             (DATA(10), CLK AFTER DELAY 10 N*SEC TO 100 N*SEC)
             WITH --------------
             CNX --------------;


CHANGE STATE OF DSSD.DATA TO SV.DATA,
                DSSD.CLK TO BIN '1'
                         AFTER_DELAY = 50 N*SEC ;
```

When the CHANGE statement is executed, the above statements cause resource set DSSD.DATA to change value to SV.DATA at the time of 'change' command (or t-zero). The resource DSSD.CLK is changed to the value of BIN '1' after a delay of 50 nanoseconds from the time of the 'change' command (or t-zero+50).

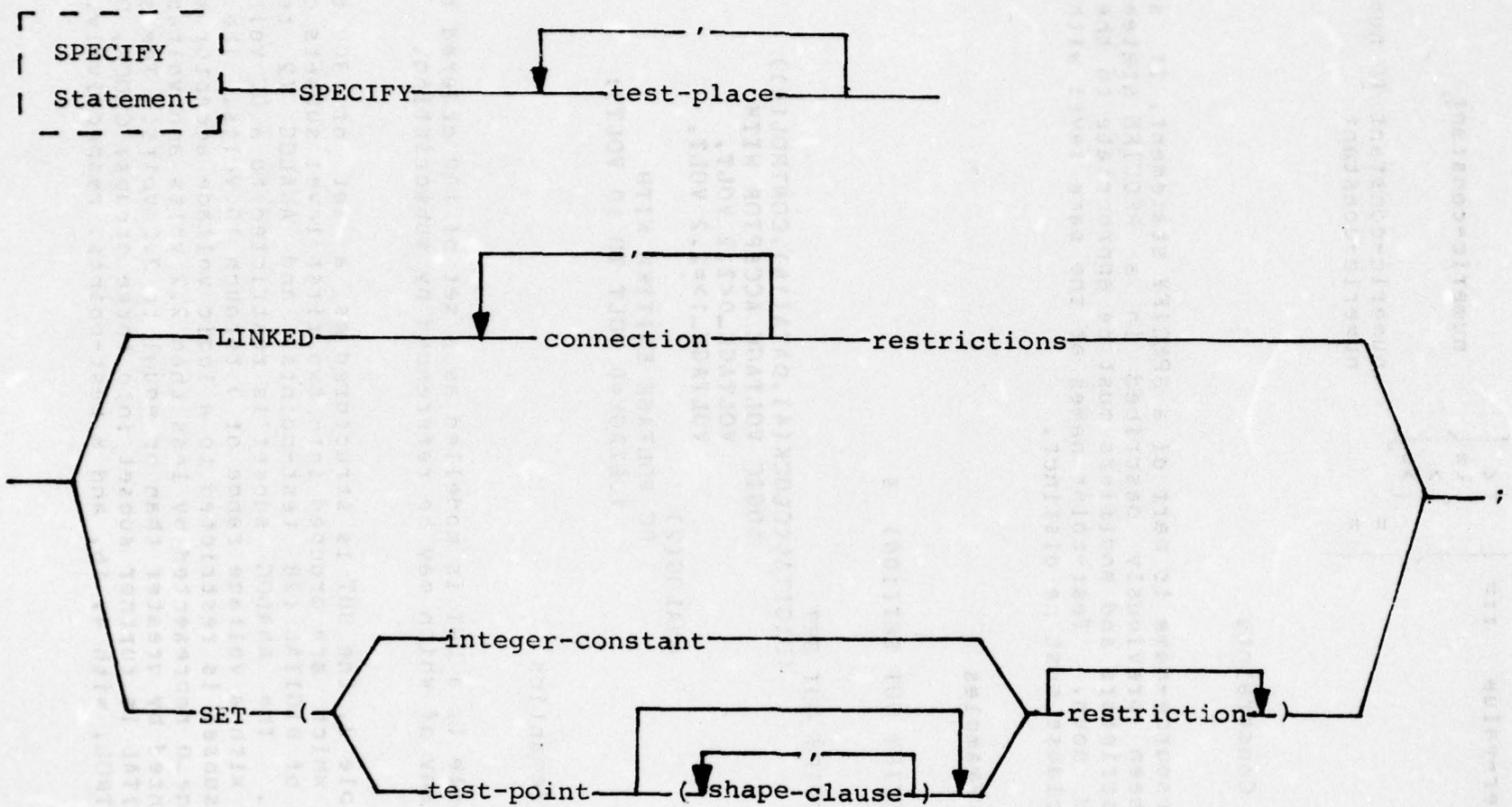
5.2.4.4 SPECIFY STATEMENT

A SPECIFY statement describes an actual test-point on a UUT, and the constraints, if any, to be imposed on that point.

5.2.4.4.1 Syntax

```
specify-stmt   ::= SPECIFY test-place,... { linkage restrictions } :
                                          { shape               }

linkage        ::= LINKED connection,...

connection     ::= pin-descriptor = test-point

pin-descriptor ::= name

test-point     ::= name

shape          ::= SET (shape-clause)

shape-clause   ::= { integer-constant                    } [restrictions]
                   { test-point [(shape-clause,...)]     }

restrictions   ::= AS { { resource-name } { EMITTER  }   specify-spec }
                      { { noun modifier } { ACCEPTOR }                 }

                      ORAS...

specify-spec   ::= WITH limitation,...

limitation     ::= modifier [modifier-value],...
```

SPECIFY STATEMENT



```
modifier-value   ::=  |  { <  }
                      |  { <= }   numeric-constant
                      |  { >  }
                      |  { >= }
                      |  =        numeric-constant TO numeric-constant
                      |  =        numeric-constant
```

5.2.4.4.2 Constraints

If a resource-name is part of a SPECIFY statement, it shall have been previously described in a REQUIRE statement. Pin-descriptors and modifiers must be appropriate to the indicated noun. Test-point names at the same level within a shape-clause must be distinct.

5.2.4.4.3 Examples

1) SPECIFY UUT SET(100) $

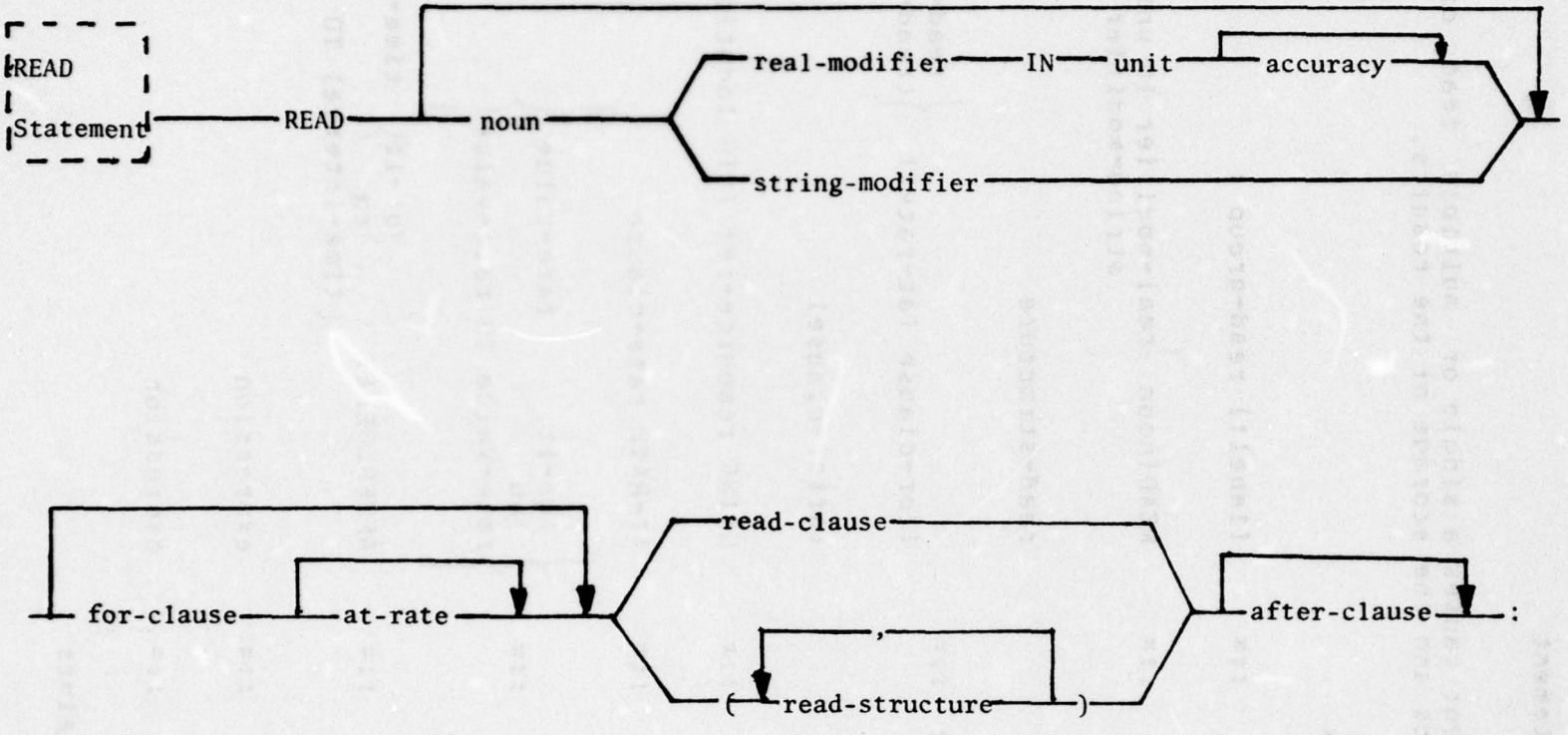
2)
```
SPECIFY UUT SET
          (DIGITAL(CLOCK(4),DATA(16),CONTROL(8))
                  LOGIC VOLTAGE ACCEPTOR WITH
                        VOLTAGE_0<2.2 VOLT,
                        VOLTAGE_1>=2.2 VOLT,
           ANALOG(2)
                  DC VOLTAGE EMITTER WITH
                    VOLTAGE=0 VOLT TO 10 VOLT);
```

5.2.4.4.4 Semantics

In Example 1, a UUT is modelled as a set of 100 ordered test points any of which may be referenced by subscripting.

In Example 2, the UUT is structured as a set of 30 test points which are grouped into two first level subsets consisting of DIGITAL (28 test-points) and ANALOG (2 test-points). The ANALOG subset is restricted to a DC voltage emitter with a voltage range of 0 through 10 volts. The DIGITAL subset is restricted to a logic voltage acceptor with a voltage_0 represented by less than 2.2 volts and voltage_1 represented by greater than or equal to 2.2 volts. The subset DIGITAL is further subset into three groups; CLOCK, DATA and CONTROL, with 4, 16, and 8 test-points, respectively.

READ STATEMENT



5.2.4.22 READ Statement

A READ statement causes a single or multiple read of UUT characteristics and the storage of the results.

5.2.4.22.1 Syntax

```
read-stmt        ::=  [label:] read-group ;

read-group       ::=  READ[noun {real-modifier IN unit[accuracy]
                                 string-modifier}
                        read-structure

read-structure   ::=  [for-clause [at-rate]] {read-clause
                                              (read-structure,...)}
                        [after-clause]

read-clause      ::=  USING resource-ref INTO location

at-rate          ::=  AT-RATE rate-clause

rate-clause      ::=  {{gt-lt} rate-value
                       {eq}
                      rate-value TO rate-value}

after-clause     ::=  AFTER_DELAY {{gt-lt} time-interval
                                   {eq}
                                   time-interval TO time-interval}

rate-value       ::=  expression

time-interval    ::=  expression
```

5.2.4.22.2 Constraints

The named noun and modifier, if used, must be the same as those in the REQUIRE statement. If more than a single resource is referenced in the iterative or recursive structure, they must all be components of the same base set. If the at-rate clause is utilized, the data-rate modifier must have appeared as a modifier in the REQUIRE statement. If the after-clause is utilized, that clause must have appeared within the resource set definition.

5.2.4.22.3 Examples

```
1)    READ AC VOLTAGE IN M*VOLT USING MVA-1 INTO VI ;

2)    READ DC VOLTAGE IN VOLT (USING DVM INTO V1,
                               USING DVM INTO V2)
                                 AFTER-DELAY = 100 M*SEC ;

3)    READ DC VOLTAGE IN VOLT
            FOR I=1 TO 10 AT_RATE = 1 K*HZ
                (USING DVM INTO V(I))  ;

4)    READ DC_LOGIC STATE
            FOR I=1 TO 8 AT_RATE = 10 K*HZ
                    USING DSSR.S INTO RW:I:  ;
```

Explanations

1) Read the value measured by the resource MVA-1; scale the value into volts, if necessary, and store the result in location VI.

2) Read the value measured by the resource MVA-1; scale the value into volts, if necessary, and store the result in location V1. After a delay of 100 milliseconds repeat the operations but store the result in V2.

3) Read the value measured by the resource DVM; scale the value into volts, if necessary, and store the result in the elements (1 through 10) of the array V. The 10 readings shall be made at a rate of 1 kilohertz.

4) The digital logic sensor DSSR.S will read an 8-bit serial data item at a bit rate of 10 kilohertz and store the result into the bit-string data word RW such that the first bit received becomes bit 1 of the word, (i.e., most significant bit first).

5.2.4.22.4 Semantics

The current value(s) being sensed by the named resource is recorded, scaled into the specified units, if necessary, and stored in the specified location.

5.2.4.26 CHANGE STATEMENT

This statement is used to cause a specified resource characteristic to be changed.

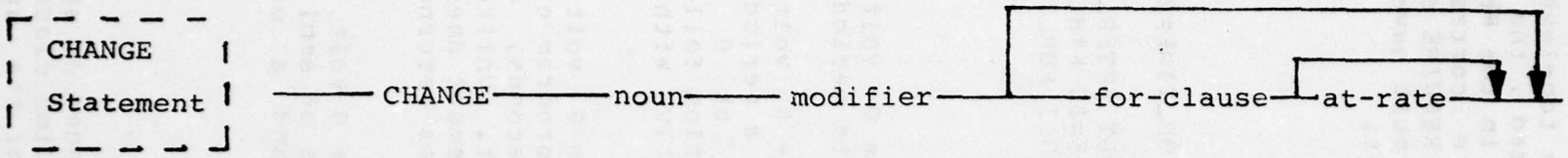
5.2.4.26.1 Syntax

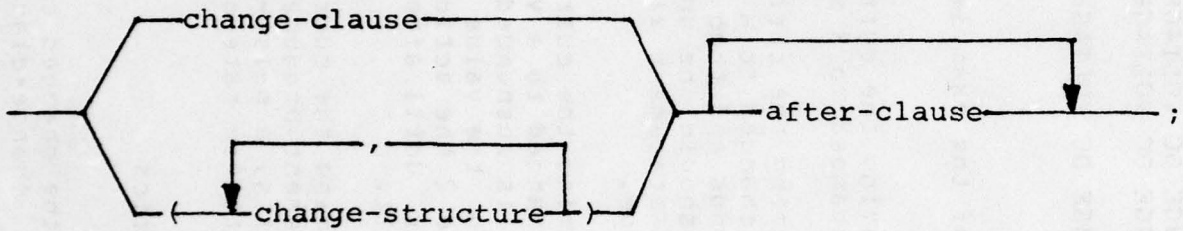
```
change-stmt       ::=   [label:] change-group ;

change-group      ::=   CHANGE noun modifier change-structure,...

change-structure  ::=   [for-clause [at-rate]]  { change-clause
                                                { (change-structure,...)
                        [after-clause]

change-clause     ::=   OF resource-ref TO value [for-time]

at-rate           ::=   AT_RATE rate-clause

rate-clause       ::=   { { gt-lt } rate-value
                        { { eq   }
                        { rate-value TO rate-value }

rate-value        ::=   expression

after-clause      ::=   AFTER_DELAY time-clause

for-time          ::=   FOR_TIME time-clause

time-clause       ::=   { { gt-lt } time-interval
                        { { eq   }
                        { time-interval TO time-interval }

resource-ref      ::=   name. ...[:component-selector:]

component-selector ::=  integer-constant
```

5.2.4.26.2 Constraints

The named modifier must have been associated with the resource in a REQUIRE statement. If more than a single resource is referenced in the iterative or recursive struc-

CHANGE STATEMENT





tures, they must all be components of the same top-level set of resources. If the at-rate clause is utilized , the rate modifier must have appeared as a modifier in the REQUIRE statement. If the after-clause is utilized, a corresponding after-clause must have appeared in the REQUIRE statement. If the for-time clause is utilized, it must have also appeared as a modifier in the REQUIRE statement.

5.2.4.26.3 Examples

```
1)  CHANGE AC VOLTAGE OF ACS TO 8 VOLT;
2)  CHANGE DC VOLTAGE OF DCS1 TO 4 VOLT FOR_TIME=2 SEC;
3)  CHANGE DC VOLTAGE (OF DCS1 TO 4 VOLT,
                       OF DCS1 TO 0 VOLT AFTER_DELAY=2 SEC);
4)  CHANGE DC VOLTAGE FOR I=1 TO 8 AT RATE=10 K*HZ
                       (OF DCS1 TO 4 VOLT FOR_TIME=1 U*SEC);
```

An analysis of the examples follows:

1) Assuming the current value of ACS were 0 volt, it is changed to 8 volts for an indefinite period.

2) Assuming the current value of DCS were 0 volt, it is changed to a value of 4 volts for a period of 2 seconds and then returns to the value of 0 volt. It should be understood that the action following that statement will occur concurrently with the change.

3) Assuming the current value of DCS were 0 volt, it is changed to a value of 4 volts and program execution is suspended for a period of 2 seconds, after which the value is returned to 0 volt. Unlike example 2 the action following the statement does not occur until after the value of DCS has returned to 0 volt.

4) Assuming the current value of DCS were 0 volt, this statement produces a burst of 8 pulses of amplitude 4 volts, a pulse-width of 1 usec, and a pulse-repetition rate of 10 KHZ.

5.2.4.26.4 Semantics

The value of the changed characteristic is changed as specified by the change-clause(s). If the for-time clause is utilized, the change occurs for the duration of the specified time after which the characteristic returns to the original value. It should be noted that the action following such usage occurs concurrently, or as nearly so as the system will allow. If the after-clause is utilized, it refers to the preceding clause and implies a delay from the CHANGE command or, if the resource reference is a sub-set then the delay is from the CHANGE of the next outermost set (or sub-

set). If the for-clause is utilized, the following group is iterated as specified at either a rate controlled by the rate-clause or of an unspecified rate. If unspecified, the rate is assumed to be as fast as the system will allow.

This Appendix contains the nouns, modifiers, and units of measure which form a major part of the OPAL vocabulary.

The Appendix consists of two major sections, and a source reference list, as follows:

Section I - Electronic-Avionics
Section II - Mechanical, Pneumatic-Hydraulic, and Optical-Radiometric
Source Reference List

Each section contains a noun and a modifier list. For Units, see Appendix 2. Each language element is shown in its OPAL form (excessively long words are abbreviated) as well as its full form.

Noun lists consist of each noun, its definition, and the modifiers which fall within that class of characteristics. The full modifier name, its OPAL form, and the units in which each may be measured are included. Units are shown in their OPAL form only. When applicable, the noun list begins with a set of modifier suffixes. Suffixes are defined as terms which are combined with a modifier to more precisely describe a characteristic.

Modifier lists contain the definition of each characteristic, a source reference, and the nouns with which each modifier is used.

The source reference list contains the bibliography from which the definitions were drawn.

The vocabulary is designed to be expandable in its dictionary of nouns, modifiers, and units. As the need for additional descriptors of measurable characteristics arises, these will be submitted through proper channels and officially added to this Appendix.

SECTION I - ELECTRONICS-AVIONICS NOUN LIST

A. Suffixes:

1. Modifier Suffixes

All non-DC measurements specified in amperes or volts (e.g., current, voltage, etc.) are assumed to represent a distortionless RMS value of the named characteristic, unless one of the following suffixes is specified with the modifier:

TRMS (True Root Mean Square): The square root of the average of the square of the value taken throughout one period.

AV (Average): The value averaged over a full cycle.

PP (Peak to Peak): The total value measured over one full cycle, from the positive-going peak to the negative-going peak.

P (Peak): Designates the instantaneous value of the positive-going peak of a time function.

P_Neg (Peak-Negative): Designates the instantaneous value of the negative-going peak of a time function.

0 (Zero) and 1 (One): Modifier suffixes to be used with the modifiers defining the logical state to describe the analogue characteristics which represent a 0 and 1 respectively in the STATE bitstring.

Note 1: The value of _P and _P_NEG shall be designated as negative with a minus sign or as positive with either no sign or a plus sign.

Note 2: A break character shall be used to join a suffix with a modifier as:

VOLTAGE_AV

2. Noun Suffix

To enable any characteristic to represent logical data the suffix LOGIC will be used with any applicable noun. The use of the suffix implicitly defines that there is a logical state bit-string modifier associated with the resource. To define the logical states requires that the modifier appear twice, with 0 and 1 suffixes, in both the resource definition and the appropriate executable statements.

The use of a digital logic resource may require the use of additional modifiers to the set associated with the noun, these are summarized below:

1)	STATE	Modifier to describe the logical 'state' of a digital logic resource set.
2)	DATA_RATE	Describes, for sources, the rate at which CHANGEs can occur, and for sensors the rate at which READings can be made, normally has an HZ dimension. In general, this modifier must be CONTROLable, and is programmed by the at_rate clause in the CHANGE and READ statements, for source and sensor resources, respectively. However, if the usage within an entire test program requires that the value is constant then it may appear as a CAPABILITY modifier.
3)	DATA_WIDTH	Used only with source resources and describes the pulse width, in units of time for a single 'bit' of data. In general, this modifier must be CONTROLable and is programmed by the for-time clause appearing in the change statement. However, if the usage within an entire test program requires that the value is constant then it may appear as a CAPABILITY modifier.

4) BUFFER_SIZE

The assumed model of a digital resource for dynamic testing includes a memory buffer. The dynamic testing construct proposed describes the number of 'bits' transmitted or received, at the specified rate, in a contiguous stream. In order to determine if a specific actual resource can satisfy the virtual resource it is necessary for the maximum number of bits issued in a continuous stream, within the entire test program be specified as a CAPABILITY BUFFER_SIZE modifier in the virtual resource REQUIRE statement. It should be understood that if the actual resource BUFFER_SIZE is less than the virtual, it does not necessarily preclude the possibility of the REQUIREMENT being satisfied. This is so since the maximum REQUIRE DATA_RATE may be such that buffering is not required.

5.3.4 Appendix 4 - Verbs and Keywords

```
verb       ::= DECLARE| DEFINE| REQUIRE| SPECIFY|
               PARTITION |SET |IF| REPEAT |LEAVE| CYCLE|
               CALL| RETURN| ESCAPE |TABLE| START|
               CHANGE| READ |DELAY |OUTPUT |INPUT|
               APPLY |REMOVE| MEASURE| ACTIVATE |GOTO|
               ENABLE| DISABLE |TERMINATE| SETUP |CONNECT|
               DISCONNECT |CLOSE |OPEN |BEGIN |MONITOR |
               REFERENCE |INCLUDE |BEGIN_BLOCK |CHAIN|
               COMMON | RUN

keyword    ::= TRUE| FALSE| BIN |OCT |HEX |BCD| XOR |OR |
               AND |NOT| ROTATE_LEFT |ROTATE_RIGHT |
               SHIFT_LEFT |SHIFT_RIGHT |ARITH_SHIFT_LEFT |
               ARITH_SHIFT_RIGHT |IN |IS |INTEGER |REAL|
               BOOLEAN |CHARSTRING |BITSTRING |ARRAY |SHARE|
               INITIAL |FUNCTION |SUBROUTINE |INTERRUPT |TASK |
               PRIORITY |RESULT |END |CHECKING| EVERY |PC |
               CNX |SOURCE |SENSOR |LOAD |CLOCK |EMITTER |TO |
               ACCEPTOR |AS |ELSE| THEN| FOR |BY |WHILE |UNTIL |
               DO |FROM |INTO |BUT |AFTER| USING |FORMAT|
               AT |OPAL| PROGRAM |MANUAL_INTERVENTION |
               INPUT_D |OUTPUT_D |WHEN |WITH |FULL_SCALE |
               READING |SET |AFTER_DELAY |CONTROL |
               CAPABILITY |LIMIT |LINKED |ORAS |FOR_TIME |
               AT_RATE |OF
```

Revised 1 February 1977

TMDE DISTRIBUTION

ARMY

Commander
U.S. Army Materiel Development
and Readiness Command
5001 Eisenhower Avenue
Alexandria, VA 22333

1 ATTN: DRCDL - J. A. Bender
1 ATTN: DRCDE-T
1 ATTN: DRCDE-DS

Commander
U.S. Army Tank Automotive
Materiel Readiness Command
Warren, MI 48090

1 ATTN: DRSTA-RGD
1 ATTN: DRSTA-MST - Mr. R. Watts

Commander
U.S. Army Aviation Systems Command
P.O. Box 209
ATTN: DRSAV-FPP
St. Louis, MO 63120

Commander
U.S. Army Electronics Command
Ft. Monmouth, NJ 07703

1 ATTN: DRSEL-SA - Mr. R. Caccamise
1 ATTN: DRSEL-TL-M- Mr. M. Tenzer
1 ATTN: DRSEL-MA-D- Mr. J. Carter
1 ATTN: DRSEL-NL-BG Dr. E. Lieblein

Commander
Tobyhanna Army Depot
Tobyhanna, PA 18466

1 ATTN: DRXTO-MI-O - Mr. J. Buckland
1 ATTN: Dep. Dir, Mr. W. Morris

Commander
U.S. Army Materiel Systems
Analysis Agency
ATTN: DRXSY-CC - Mr, D. Lynch
Aberdeen Proving Ground, MD 21005

Commander
U.S. Army Missile Command
Huntsville, AL 35809

1 ATTN: DRSMI-RLD - Mr. D. Thurman
1 ATTN: DRSMI-RLE - Mr. G. Hubbard
1 ATTN: Director of Maintenance, TMDE Ofc

Commander
Army Satellite Communications Command
ATTN: DRCPM-SC-8B
Ft. Monmouth, NJ 07703

Commander
U.S. Army Maintenance Management Center
ATTN: DRXMD-TR - Mr. P. Smith
Lexington, KY 40507

Commander
U.S. Army ARADMAC
ATTN: SAVAE-EMP - Mr. L. Baca
Corpus Christi, TX 78419

Army Security Agency
ATTN: IALOG-RME
Arlington, VA 22212

Commander
Sacramento Army Depot
Sacramento, CA 95813

1 ATTN: DRSXA-MME - Mr. H. Kuyama
1 ATTN: Dep. Dir, Maint - Mr. A. Weaver

Commander
Picatinny Arsenal
ATTN: TSD - Mr. D. Morlock (Bldg 352)
Dover, NJ 07801

Commander
U.S. Army Logistics Center
ATTN: ATCL-MC - Mr. C. Adenauer
Ft. Lee, VA 23801

DISTRIBUTION

ARMY (continued)

Commander
TRASANA
ATTN: ATAA-TDL
White Sands Missile Range, NM 88002

DA-ODCSLOG
ATTN: DALO-SMM-E - Mr. Nichols
Washington, DC 20310

COL E. A. Viereck, Jr.
ASA L&L
Room 3E619
Pentagon Building
Washington, DC 20310

PM, ATSS
ATTN: DRCMP-ATSS, LTC J. Gabrysiak
Ft. Monmouth, NJ 07703

Commander
U.S. Army Troop Support Command
4300 Goodfellow Boulevard
St. Louis, M) 63120

Commander
U.S. Army Armament Command
ATTN: DRSAR-RDG
Rock Island, IL 61201

Headquarters, DA
DAMA-CSS
Washington, DC 20310

PM, ARTADS
ATTN: DRCPM-TDS-TF - Mr. David Blank
Ft. Monmouth, NJ 07703

Commander
U.S. Army TARADCOM
ATTN: DRDTA-RGD - Mr. J. Steyaert
28251 Van Dyke
Warren, MI 48090

DISTRIBUTION

NAVY

Headquarters
Naval Materiel Command
ATTN: Director, Automatic Test Equipment Management & Technology Office (MAT 036T)
Washington, DC 20360

Commander
Naval Air Systems Command
ATTN: AIR 53424, Mr. M. Myles
Washington, DC 20360

Commander
Naval Electronic Systems Engineering Center
ATTN: 03T, Mr. F. Fernandez
P.O. Box 80337
San Diego, CA 92138

Commander
Naval Electronic Systems Command
ATTN: ELEX 4802, Mr. G. Margulies
Washington, DC 20360

Commander
Naval Sea Systems Command
ATTN: SEA 9822B, Mr. John Yaroma
Washington, DC 20360

Commander
Naval Electronics Laboratory Center
ATTN: Code 4050, Mr. D. Douglas
271 Cataline Boulevard
San Diego, CA 92152

DISTRIBUTION

AIR FORCE

Commander
Kelly Air Force Base
San Antonio, TX 78241

1 ATTN: SA-ALC/MMIN-Mr. J. Ferrell
1 ATTN: SA-ALC/XRXM-Mr. T. Daily
1 ATTN: SA-ALC/MMEC-Mr. J. Cotnam
1 ATTN: SA/ALC/MAGT-Mr. H. Arant

Headquarters
AFSC/LGM
Andrews Air Force Base
ATTN: Mr. Clark Walker
Washington, DC 20334

Headquarters
USAF/LGYE
ATTN: Mr. C. Houk
Washington, DC 20330

Commander
Wright Patterson Air Force Base
Dayton, OH 45433

1 ATTN: ASD/ENEGE-Mr. D. Behymer
1 ATTN: ASD/AEG

DISTRIBUTION

OTHER

Defense Documentation Center (12)
Cameron Station
Alexandria, VA 22314

NASA
J.F. Kennedy Space Center
ATTN: DE-DEO-2, Mr. L. Dickison
Kennedy Space Center, FL 32899

RLG Associates, Inc.
11250 Roger Bacon Drive
Suite 16
Reston, VA 22090

Ralph Shirak
RCA Corporation
P.O. Box 588
Burlington, MA 01801

B. Richard Climie
Aeronautical Radio, Inc.
2551 Riva Road
Annapolis, MD 21401

Roger W. Fulling
Hughes Aircraft Company
1515 Wilson Boulevard
Arlington, VA 22209

Leon G. Stucki
McDonnell Douglas Astronautics Co.
5301 Bolsa Avenue
Hungtington Beach, CA 92647

Dr. N. S. Prywes
Dept. of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19174

Center for Computer Sciences & Technology
National Bureau of Standards
Washington, DC 20234

Massachusetts Computer Associates
26 Princess Street
Wakefield, MA 01880

1 ATTN: Mr. D. Loveman
1 ATTN: Mr. Thomas Cheatham, Jr.

Robert G. Kurkjian
Hughes Aircraft Company
Bldg 6, MS E111
Culver City, CA 90230

Andy Mills
Hewlett-Packard
974 East Arques
Sunnyvale, CA 94806

M. T. Schloesser
Lockheed-California Company
Div. of Lockheed Aircraft Corp., Bldg 167
Burbank, CA 91520

Analytics
2500 Maryland Road
Willow Grove, PA 19090

1 ATTN: Mr. S. Leibholtz
1 ATTN: Mr. R. Brachman

DISTRIBUTION

FRANKFORD ARSENAL

Commander
Frankford Arsenal
Philadelphia, PA 19137

1 ATTN: CRF 107/1

1 ATTN: TD 107/1

1 ATTN: PA 107/2

1 ATTN: GC, 28/1

1 ATTN: QAA-R 119/2

1 ATTN: PD, Mr. J. Corrie 64/4

2 ATTN: TSP-L 51/2

1 ATTN: FCF-C 202/1

40 ATTN: FCF 201/1

1 ATTN: FCF-S 201/1

1 ATTN: FCF-E 201/1

6 ATTN: TSP-T 51/2